Tiago André Gonçalves Castanho

Plataforma Epik - Componente gráfica e interativa para criação de fluxos de cenários

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Orientadora: Fernanda Barbosa, Profª. Auxiliar, Universidade Nova de Lisboa

Co-orientadora: Carmen Morgado, Profª. Auxiliar, Universidade Nova de Lisboa

Júri

Presidente: Prof. Doutor Jorg Matthias Knorr
Arguente: Prof. Doutor Carlos Jorge de Sousa Gonçalves
Vogal: Profª. Doutora Fernanda Maria B. T. Vieira Barbosa

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Julho, 2019

Plataforma Epik - Componente gráfica e interativa para criação de fluxos de cenários

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Agradecimentos

Gostaria de aproveitar esta página do documento para agradecer às pessoas que influenciaram o meu percurso académico no Mestrado Integrado em Engenharia Informática.
Aproveitar para agradecer às minhas orientadoras, Profª Fernanda Barbosa e Profª Carmen Morgado, que sempre nos motivaram para fazer mais e melhor com o objetivo de ser entregue o melhor trabalho possível.
Agradecer aos meus pais por toda a paciência e compreensão que demonstraram nestes últimos 5 anos.
À Inês, que durante estes anos esteve sempre do meu lado nos bons e maus momentos fazendo, simultaneamente, o seu percurso académico e sendo um exemplo a seguir.
Ao melhor grupo de amigos que podia ter arranjado nesta fase da vida que souberam não só lidar com momentos felizes mas também frustrações, sendo um exemplo de companheirismo e entreajuda: André Catela, Duarte Cruz, Francisco Cardoso, João Santos e Ricardo Fernandes.
Por último, uma palavra de agradecimento ao Henrique Pina e a todos os meus atletas nestes últimos dois anos que sem saberem foram importantes para mim como exemplo de trabalho e organização tanto no desporto como na vida.

Resumo

Hoje em dia, com o aumento da desmotivação dos alunos foi necessário encontrar soluções para os manter interessados no ensino. Após estudos de vários filósofos e psicólogos podemos concluir que os jogos, quando bem aplicados ao ensino obtêm bons resultados, tanto a nível intelectual como a nível social. No caso de serem mal aplicados, os jogos não são produtivos e os alunos jogam sem reter nenhum conhecimento. Caso os jogos sejam colaborativos, os alunos estão mais predispostos a receber tarefas mais complexas, ouvir outras ideias e trabalhar em equipa. Outras das vantagens da colaboração são a inclusão de alunos com necessidades especiais e o desenvolvimento de *soft-skills*.

A plataforma Epik é uma ferramenta para desenvolver jogos sem ter a necessidade recorrer à programação para a criação do jogo. Os jogos Epik são um exemplo de jogos colaborativos para a educação que estimulam a entreajuda. No entanto, na sua versão anterior a plataforma tinha algumas falhas, como por exemplo: Todos os jogadores são obrigados a estar no mesmo cenário em simultâneo; Falta de variedade nos mecanismos de transição de cenários; Falta de uma componente gráfica que permitisse obter uma visão geral do fluxo criado.

Esta dissertação propõe solucionar estes problemas implementando uma componente gráfica que permita, aquando do desenvolvimento do jogo, definir as diferentes possibilidades de transitar entre cenários, possibilitando a definição de diferentes percursos na execução do jogo de acordo com o tipo de critério definido pelo professor. Será também implementada uma nova funcionalidade que permita aos jogadores, durante a execução do jogo, visualizem o percurso já efetuado e o estado dos seus colegas de jogo, no caso de jogos *multiplayer*. Assim sendo, o Epik foi reformulado de forma a colmatar as falhas na anterior versão da plataforma, permitindo a mesma criação de jogos educativos através de uma interface gráfica mas neste caso, durante a execução jogo, permitindo que cada jogador seja independente não sendo obrigado a estar no mesmo cenário que os restantes jogadores e encontrando cenários mais adequados ao seu nível de desempenho.

Palavras-chave: jogos educativos; fluxos de execução; colaboração; plataforma Epik;

Abstract

Nowadays with the increase of students demotivation it was necessary to find solutions to keep them interested in the education. After several studies of philosophers and psychologists we can conclude that games, when well applied to teaching, obtain good results, both at intellectual and social levels. In case they are misapplied, the games are not productive and the students play without retaining any knowledge.
If the games in question are collaborative, students are more predisposed to receive more complex tasks, listen to other ideas and work as a team. Other advantages of collaboration are the inclusion of students with special needs and the development of soft skills.
The Epik platform is a tool to develop games without having to resort to programming to create a game. The Epik games are an example of collaborative games for education that encourages mutual support. However, in the older version, the platform had some flaws such as: All players are required to be in the same scenario simultaneously; Lack of variety in scenario transition mechanisms; Lack of tab to sort the scenarios and have a better perspective of the game as a whole.
This dissertation proposes that in order to solve these problems it will be implemented a graphical component that allows, during the development of the game, to define the different possibilities of going through scenarios. Which will allow the definition of different routes in the game defined by the professor. It will be implemented a new functionality which allows the players who are playing the game see their completed route and their colleagues status if the game is in multiplayer mode. Therefor, the Epik has been redesigned to resolve the flaws in the previous version of the platform, allowing the same creation of educational games through a graphical interface but in this case, during the execution game, allowing each player to be independent not being forced to be in the same scenario than the remaining players and thus finding scenarios more suited to their level of performance.

Keywords: educational games; execution flow; collaboration; Epik platform;

Índice

Lista de Figuras

Lista de Tabelas

Listagens

Siglas

CSS	Cascading Style Sheets.
Epik	Edutainment by Playing and Interacting with Knowledge.
HTML	HyperText Markup Language.
JSON	JavaScript Object Notation.
PDF	Portable Document Format.
URL	Universal Resource Locator.
VB	Visual Basic.
VPN	Virtual Private Network.
XML	Extensible Markup Language.

Capítulo 1

Introdução

Este trabalho tem como objetivo reformular a plataforma Epik, dar aos utilizadores uma interface gráfica de modo a poder criar fluxos de jogo de forma interativa e aos jogadores uma maior flexibilidade durante a execução do jogo removendo a limitação existente que obriga todos os jogadores estarem no mesmo cenário no caso dos jogos multiplayer.

1.1 Motivação

Os jogos são usados no ensino como atividades educativas mais cativantes para os alunos de forma a motivá-los para o ensino, sendo que hoje em dia, já existem muitos tipos jogos educativos que se podem usar em determinadas áreas como por exemplo: jogo da memória, forca, palavras cruzadas e sudoku.
No âmbito da educação, a maioria dos jogos utilizados nos dias de hoje são individuais. Todavia, quando existe colaboração, os alunos estão dispostos a aceitar tarefas mais complexas devido ao compromisso gerado dentro do grupo de trabalho e para além disso, existe uma integração mais fácil, incluindo os alunos com necessidades especiais criando um ambiente de entre-ajuda onde se partilha conhecimentos e opiniões entre os elementos do grupo. Ainda que a colaboração seja um meio para motivar a aprendizagem, existem poucos jogos educativos que utilizem os mecanismos colaborativos.
O desenvolvimento de jogos colaborativos não é uma tarefa fácil na maioria das plataformas de desenvolvimento de jogos porque a maioria exige conhecimentos prévios de programação, sendo exemplos disso o Construct [1], Unity [2] e Game Maker: Studio [3]. De modo a ultrapassar algumas destas dificuldades foi implementada a plataforma Epik que permite a criação de jogos educativos colaborativos através de uma interface gráfica e é composta por dois servidores distintos: servidor de desenvolvimento de jogos e servidor

de execução de jogos. Os jogos Epik são compostos por vários cenários, que poderão ser jogados no modo *singleplayer* ou no modo *multiplayer*, e a ligação entre estes é denominada de fluxo de cenários. Atualmente, não é possível obter uma visão geral deste fluxo de cenários, quer seja no momento de desenvolvimento do jogo ou no momento de execução do mesmo, algo que pode desmotivar os utilizadores e os jogadores no caso de estarem perante um fluxo mais complexo. Estes jogos, no modo de jogo *multiplayer* são colaborativos, isto é, pode existir entreajuda nas atividades entre os alunos que estão a jogar, no entanto, a evolução no jogo é igual e em simultâneo para todos os jogadores. Este tipo de jogo já foi usado em situação de sala de aula com resultados positivos ao cativar a atenção dos alunos e, para além da sua utilização na sala de aula, também serve de repositório de recursos, que podem ser importados nos formatos de áudio, vídeo, PDF ou imagem, e atividades educativas para os alunos aplicarem os seus conhecimentos. A plataforma Epik mantinha *online* os resultados e os *rankings* dos diferentes jogos no servidor de execução de jogos e apesar de das suas qualidades ainda existem vários aspetos a melhorar na plataforma. Nos jogos, as atividades são apenas questões de verdadeiro ou falso, escolha múltipla ou resposta curta, o que torna o mecanismo de aprendizagem repetitivo e limitado pois existem alunos que não se motivam com este tipo de atividade e para além disso, existem planos curriculares onde outro tipo de atividade é mais adequado para a aprendizagem por parte dos alunos.
A evolução no jogo não depende do nível de conhecimento do jogador e é sempre feita em simultâneo entre todos do jogadores (quando em modo *multiplayer*), o que pode levar à desmotivação da execução das atividades por parte dos jogadores, pois ao terminar um cenário, o jogador terá que esperar que todos os seus colegas o terminem para que possam evoluir no jogo. Para melhorar este aspeto terão que ser redefinidos os mecanismos de evolução já existentes de modo a cada jogador evoluir no jogo como individuo e com base no seu desempenho, ainda que com o espírito de entreajuda. Logo, a necessidade de ter vários fluxos de execução e independentes nos jogos *multiplayer* e a facilidade de visualizar esses possíveis fluxos de cenários será o tema central desta tese.
Outro dos requisitos, para além das melhorias já referidas nos jogos Epik, é a implementação do Epik em formato de aplicação *desktop* o que vai permitir aos utilizadores o desenvolvimento de jogos em locais sem acesso à internet e um maior controlo sobre os dados pessoais e recursos sem a necessidade destes serem guardados *online*.

1.2 Enquadramento e contexto

O projeto denominado ”Reestruturar, Flexibilizar e Atualizar a plataforma Epik” tem como objetivo implementar a nova plataforma Epik, tendo como foco principal a implementação de:

- Aplicação *desktop* para o desenvolvimento de jogos através de um ambiente gráfico;
- Servidor de jogos Epik online.

Para além disso, os novos jogos Epik deverão permitir diversidade no tipo de atividades educativas e nos mecanismos de colaboração, bem como diferentes evoluções no jogo consoante o desempenho de cada jogador. Neste projeto estão integradas cinco teses do Mestrado Integrado em Engenharia Informática que correspondem à criação de componentes integrados no Epik:

- Uma interface gráfica para a criação de fluxos de cenários num jogo Epik e especificação de novos mecanismos de evolução - **Tiago Castanho**;
- Atividades de programação que poderão ser definidas num jogo Epik - **André Correia**;
- Atividades de puzzle, podendo estes serem puzzles tradicionais, associação de conceitos ou quebra-cabeças - **André Catela**;
- Atividades de *quiz*, abrangendo uma maior variedade de tipos de questões e não apenas escolha múltipla e resposta curta como já desenvolvido - **Daniela Santos**;
- Mecanismos de colaboração para além dos já existentes na atual plataforma - **Alexandra Silva**.

1.3 Objetivos

O objetivo principal desta tese será implementar uma componente gráfica de modo a facilitar a criação de um fluxo de cenários, fazendo com que seja mais intuitivo ao utilizador criar o fluxo do jogo e abrangendo todo o nível de jogadores sem que sejam limitados a estar todos no mesmo cenário em simultâneo, algo que acontece na versão atual. Atualmente apenas existem quatro mecanismos de transição entre cenários e os utilizadores em que, ou todos os jogadores fazem *skip*, ou todos os jogadores fazem "Continuar", ou o tempo limite do cenário termina ou terminam o cenário com sucesso, tanto o que está a criar o jogo como o jogador que está a jogar, não têm como obter uma vista geral de todo o fluxo nos casos em que o jogo é maior/mais complexo, sendo o modo de construção deste fluxo apenas composto por um menu onde o utilizador seleciona os cenários de destino para cada um dos quatro mecanismos referidos anteriormente. Para melhorar esse aspeto vai ser implementada uma nova funcionalidade na plataforma que permita definir mais facilmente os possíveis fluxos de cenários de um jogo sendo que, esta componente vai permitir criar e alterar os fluxos de cenário através da construção de grafos interativos. Estes grafos são compostos por nós, que representam os cenários constituintes do jogo, e por arcos que representam as transições entre os mesmos.
Para ser possível cumprir os objetivos propostos acima é necessário, primeiro, a implementação da aplicação *desktop* Epik, sendo esta uma tarefa comum a todos os alunos envolvidos neste projeto. Os objetivos individuais desta dissertação são:

- Análise das ferramentas de visualização de grafos de modo a selecionar qual utilizar e incorporar no Epik de modo a permitir a especificação de fluxos de transição entre cenários;
- Definir e implementar novas formas de evoluir através dos cenários, de forma a permitir diferentes fluxos de execução de jogos, dependendo do conhecimento do jogador. Para além de tornar independente a evolução do jogo;
- Criar uma componente gráfica na aplicação de desenvolvimento de jogos Epik, que permita ao utilizador desenhar a sequência do jogo de forma intuitiva e fácil;
- Permitir na execução do jogo visualizar os percursos possíveis e cenários já realizados.

Para realização destes objetivos e da parte conjunta referida anteriormente deverão ser realizadas as seguintes tarefas ao longo do desenvolvimento da plataforma:

- Realizar um estudo em conjunto, sobre as ferramentas existentes para desenvolvimento de aplicações *desktop*, com o objetivo de selecionar a mais adequada para implementação da plataforma de desenvolvimento de jogos Epik;
- Implementação da nova aplicação *desktop* Epik para desenvolvimento de jogos, com o grupo de trabalho;
- Análise das possíveis formas de evolução de cenários em jogos educativos existentes, de forma a definir novos mecanismos de evolução nos jogos Epik;
- Implementação dos mecanismos de evolução de cenários na plataforma Epik;
- Análise e estudo de ferramentas que permitem criar e visualizar grafos interativos, de forma a selecionar as mais adequadas para implementar o novo separador de construção e visualização de fluxo de cenários;
- Implementação da componente gráfica na aplicação Epik para criação de cenários e seus respetivos fluxos;
- Definir em conjunto, a estrutura de um jogo gerado;
- Implementação da página Web da plataforma Epik com o grupo de trabalho;
- Desenho e implementação das áreas de Programador e Jogador com o grupo de trabalho;
- Execução dos jogos Epik permitindo aos jogadores uma evolução independente.

1.4 Organização do documento

O resto deste documento é constituído por mais seis capítulos.
No segundo capítulo é feito um levantamento de alguns trabalhos e temas relacionados com a presente dissertação, sendo também apresentada uma descrição da versão anterior plataforma Epik. As funcionalidades e descrição da nova plataforma Epik são apresentadas no terceiro capitulo, o estudo das ferramentas a utilizar para obter uma implementação com sucesso é feito no capitulo quatro e no quinto capitulo é apresentada a arquitetura da plataforma e a implementação da mesma.
No sexto e sétimo capítulos são disponibilizados os resultados dos questionários feitos aos utilizadores da plataforma e algumas ideias para trabalho futuro de forma a melhorar e atualizar a plataforma Epik.

Capítulo 2

Trabalho relacionado

Este capitulo descreve o trabalho de pesquisa relativo ao tema da dissertação, onde é feita comparação da plataforma Epik com outras plataformas de desenvolvimento de jogos, percebendo como é que os jogos podem motivar os alunos para o ensino e como os jogos Epik podem ser utilizados como ferramenta de ensino.

2.1 Jogos no ensino

Com o aumento do desinteresse e desmotivação dos alunos de hoje em dia foi necessário encontrar outras soluções para manter os alunos interessados no ensino. Assim, de maneira a desencadear interesse dos estudantes surgem as atividades lúdicas que têm como objetivo, através da satisfação, permitir a mais fácil assimilação de conceitos e conhecimentos. Segundo Piaget (1896- 1980), um psicólogo Suíço, os jogos são fundamentais para o desenvolvimento intelectual das crianças e não apenas uma formas destas gastarem energia, então, após vários estudos já realizados por bastantes psicólogos e estes se debaterem sobre o assunto como, Piaget, Vygotsky (1896- 1934) ou até mesmo Platão (427-347 a.C.), é possível chegar a conclusão que a utilização de atividades lúdicas no ensino gera entusiasmo, estimula entreajuda e obtém bons resultados diante dos alunos [4]. É possível concluir que algumas das vantagens que os jogos têm quando aplicados ao ensino são: poder relacionar várias unidades de ensino, aprender a tomar decisões, rever matérias de forma divertida, obrigar o desenvolvimento de *soft-skills* através da competição, pensamento crítico e colaboração. No entanto, também podem ter efeitos negativos quando mal aplicados pois os alunos podem estar a jogar apenas por jogar, perdendo o conhecimento essencial do jogo e consequentemente o professor acaba por perder tempo se não estiver preparado para tal situação sacrificando por vezes outras matérias.
Quanto ao tipo de jogo, estes podem classificar-se em dez categorias distintas [5]:

- **Exercícios e Prática** - o jogador responde a perguntas por reconhecimento visual, ou inserção de texto e é testado quanto à lembrança e reconhecimento de factos;
- **Jogo extrínseco de memória** - desafios com base em raciocínio não relacionado com fantasia do jogo;
- **Simulador de sistemas** - o jogador controla mais que uma personagem ou aspeto do jogo em simultâneo;
- **Simulador de personagens** - o jogador controla apenas uma (ou um número limitado) personagem e não tem controlo do ambiente em redor;
- **Jogo intrínseco de memória** - apresenta desafios que envolvem raciocínios relacionados com a fantasia do jogo;
- **Questionários de grupo** - jogos com questões onde os jogadores necessitam de interagir de forma a avançar;
- **Resolução de problemas em grupo** - jogos de memória que podem afetar os jogos dos outros jogadores;
- **Estratégia *multiplayer*** - onde as ações de diferentes jogadores influenciam o jogo de todos;
- **Interação de personagens** - simulador de personagens com o modo *multiplayer*;
- **Jogo social de memória** - jogos que envolvem a discussão de ideias entre jogadores para superar desafios.

Atualmente existem muitos sites que disponibilizam jogos educacionais para todas as idades, níveis de ensino e os mais variados temas,sendo mais comuns por exemplo os jogos de puzzle (Figura 2.1) [6] e de memória (Figura 2.2) [7].

2.1.1 Colaboração em jogos educacionais

De acordo com Gokhale (1995), a aprendizagem colaborativa dá aos alunos uma oportunidade de interagir nas discussões, responsabilidade pela própria aprendizagem e, deste modo, de se tornarem pensadores críticos [8].
A aprendizagem colaborativa não existe apenas entre professor/aluno mas sim entre todos os que fazem parte da mesma comunidade de aprendizagem e contribuem para o ensino dos demais, criando um ambiente de partilha entre toda a comunidade, sendo o método mais comum de colaboração no ensino os trabalhos de grupo onde é necessário pesquisa ou resolução de um problema, no entanto a colaboração pode também estar presente na forma de debates, jogos ou grupos de estudo.

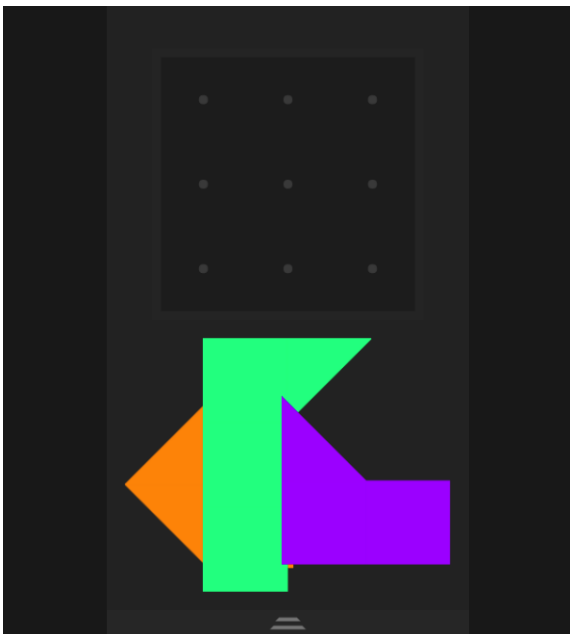


Figura 2.1: Jogo Puzzle

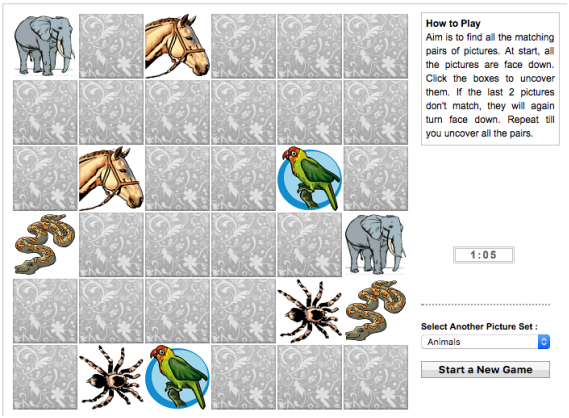


Figura 2.2: Jogo da Memória

Quando dois ou mais alunos tentam aprender/fazer algo juntos podemos definir isto como aprendizagem colaborativa [9] e usar a colaboração como ferramenta no ensino é versátil e eficaz, pois os estudantes estão mais recetivos a aceitar tarefas complexas, ouvir outras ideias e trabalhar em equipa [10]. Outra das grandes vantagens da colaboração no ensino é servir como facilitador de inclusão dos alunos com necessidades especiais e alunos mais introvertidos, e assim, os alunos serão obrigados a desenvolver *soft-skills*, partilhando conhecimentos e tendo compromissos para com os outros elementos do grupo. Hoje em dia, as *soft-skills* são cada vez mais importantes no mercado de trabalho e é necessário o ensino acompanhar essa tendência e focar também nesta ferramenta que permitirá aos alunos serem mais competentes no futuro. Também no mercado de trabalho e com o avanço da tecnologia, a colaboração está presente no dia-a-dia devido às ferramentas que permitem constante colaboração entre os membros do grupo tanto no ensino como

no meio empresarial como o Slack [11] e o Skype [12].
Porém, são poucos os jogos educativos que permitem a colaboração e um exemplo de jogos educativos colaborativos são os jogos Epik onde é estimulada a entreajuda beneficiando os jogadores que estão a ajudar.

2.1.2 Fluxos de execução em jogos educacionais

O fluxo de execução nos jogos é essencial para manter o jogador focado e interessado na tarefa que está a realizar e no caso de jogos educacionais, este fluxo torna-se ainda mais importante porque é necessário haver um equilíbrio entre o tempo despendido para ter sucesso e a dificuldade proposta. Se for muito fácil o jogador acha o jogo monótono ou se for muito difícil o jogador fica com ansiedade, e em ambos os casos deixa de jogar. Assim sendo, para o professor é importante ter controlo sobre o fluxo de cenários de modo a manter os seus alunos motivados com dificuldades adequadas ao seu conhecimento durante a execução do jogo.
Segundo Mihaly Csikszentmihalyi (1990), o fluxo tem sete componentes principais, sendo quatro pré requisitos (Tarefas claras, *Feedback*, Concentração, Objetivo alcançável) e três características que acontecem durante o jogo (Controlo, Perda da noção do tempo e Esquecimento de si próprio). Deste modo, em jogos educacionais é comum serem verificados os seguintes princípios [13].

- **Tarefas claras** - Os utilizadores entendem claramente as tarefas a realizar;
- ***Feedback*** - Existe um *feedback* imediato sobre o correu mal e o que correu bem;
- **Concentração** - Não criando distrações e os utilizadores podem executar a tarefa;
- **Objetivos alcançáveis** - O objetivo é desafiante, no entanto dentro das capacidades dos utilizadores;
- **Controlo** - Os utilizadores acreditam que as suas decisões têm impacto no desenrolar do jogo;
- **Perda da noção do tempo** - O tempo passa mais rápido, sem o utilizador ter noção disso;
- **Esquecimento de si próprio** - Tornar-se parte da atividade através de foco total na atividade.

Em alguns jogos são criados níveis com diferentes graus de dificuldade de forma a adaptar o jogo ao conhecimento do aluno, noutros são criadas sequências de atividades num jogo em que o jogo segue um fluxo linear ao longo das atividades, todavia, os jogos educativos na sua maioria são atividades isoladas, em que apenas é avaliado se o aluno respondeu corretamente, não existindo qualquer fluxo de execução ligado ao jogo.

2.2 Aplicações para desenvolvimento de jogos

Hoje em dia existe alguma variedade de aplicações que permitem aos utilizadores desenvolver um jogo, porém, o grau de dificuldade e de conhecimentos prévios exigidos é grande e algumas destas aplicações não são grátis. Uns dos exemplos mais conhecidos são:

- **Construct** - Lançado em 2007, não necessita de programação, pois toda a interface é com *drag-and-drop* mas usa *python* se o utilizador desejar complementar o seu jogo e após acabar o desenvolvimento do jogo é possível exportar para qualquer plataforma. Existe vasta documentação e tutoriais para compreender todos os conceitos do Construct, no entanto, permite apenas um número limitado de projetos a não ser que o utilizador deseje comprar uma licença pessoal. Na versão gratuita apenas permite a exportação de jogos em HTML5 perdendo a funcionalidade de multi-plataforma [1].
- **Game Maker: Studio** - A primeira versão do programa foi lançada em 1999 com o objetivo de criar animações, no entanto depois de verificar que os utilizadores usavam mais a plataforma para a criação de jogos, Mark Overmars, decidiu deixar o nome Animo e renomear a plataforma de Game Maker: Studio. No Game Maker: Studio, com apenas uma implementação é possível exportar o jogo para múltiplas plataformas e com uma interface de *drag-and-drop* permite a utilizadores sem experiência em programação um fácil desenvolvimento de jogos. O Game Maker: Studio pode também ajudar aqueles que estão interessados em aprender programação porque permite a visualização do código criado através das ações *drag-and-drop* do utilizador [3].
- **Unity** - O desenvolvimento em Unity é feito com base em componentes, dependendo de alguns conhecimentos prévios em programação, suportando o desenvolvimento em 2D e 3D e existindo vários tipos de tutoriais e documentação para permitir aos mais recentes utilizador uma rápida assimilação da plataforma. Após concluir o jogo é possível exporta-lo para várias plataformas como: iOS, Android, MacOS, Linux, Windows, Steam VR. De modo a obter todas as funcionalidades da plataforma é necessário que o utilizador compre uma licença, no entanto é possível a utilização grátis da plataforma. [2].

Na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa foi desenvolvida o Epik com o objetivo de permitir aos seus utilizadores criarem jogos educativos de forma gratuita e sem a necessidade de conhecimentos prévios na área da programação.

2.2.1 Plataforma Epik

O Epik (Edutainment by Playing and Interacting with Knowledge) é uma aplicação online dividida em dois servidores. O servidor web para desenvolvimento de jogos onde

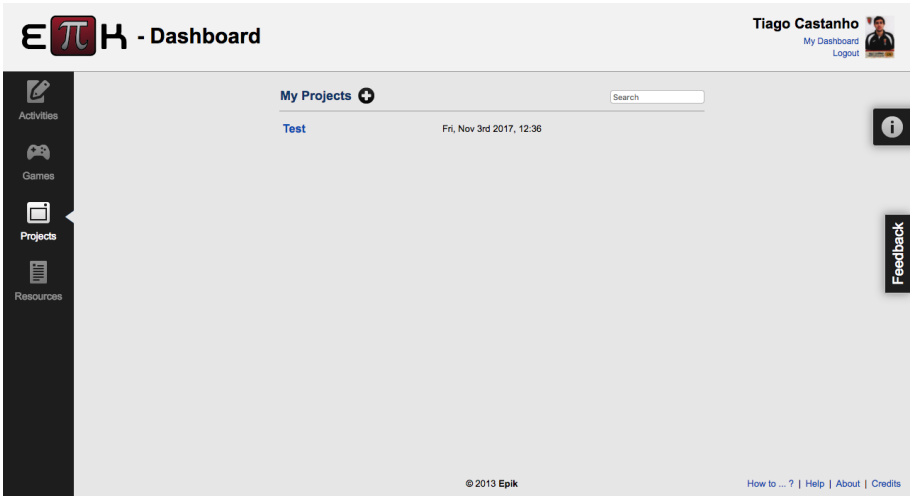


Figura 2.3: Dashboard Epik

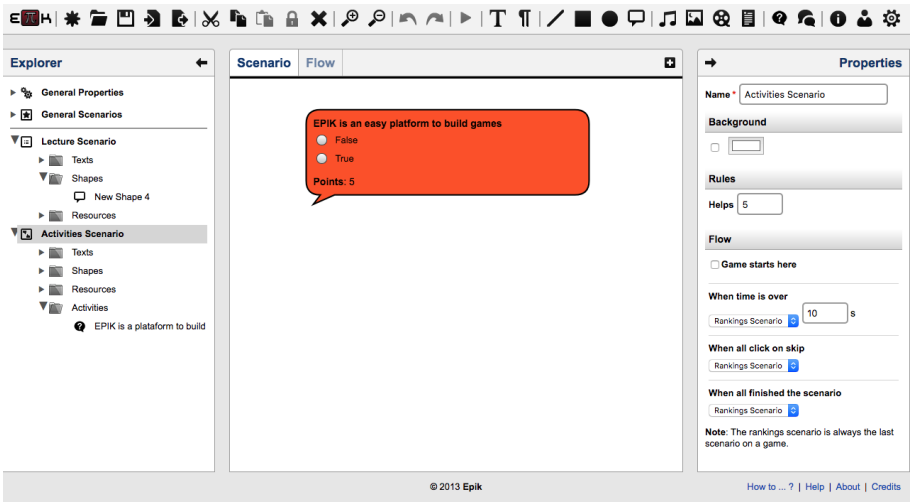


Figura 2.4: Área de trabalho Epik

os utilizadores podem facilmente criar o seu jogo e o servidor de execução de jogos em que podem ser jogados jogos Epik previamente já partilhados pelos utilizadores da plataforma.

2.2.1.1 Criação de projetos, atividades e recursos

Para criar um jogo, o utilizador deve criar um projeto na plataforma Epik de desenvolvimento de jogos, pois sem a criação do projeto apenas é possível criar atividades e recursos que após serem criados ficam disponíveis na *dashboard* (Figura 2.3) nas respetivas secções. A funcionalidade de criação de projeto está disponível na *dashboard* Epik ou na área de trabalho e para criar o utilizador necessita de escolher o modo de jogo (Questionário individual ou Questionário colaborativo) e um *template* de projeto, que pode ser um projeto em branco. Os projetos serão conjuntos de cenários compostos por atividades e recursos

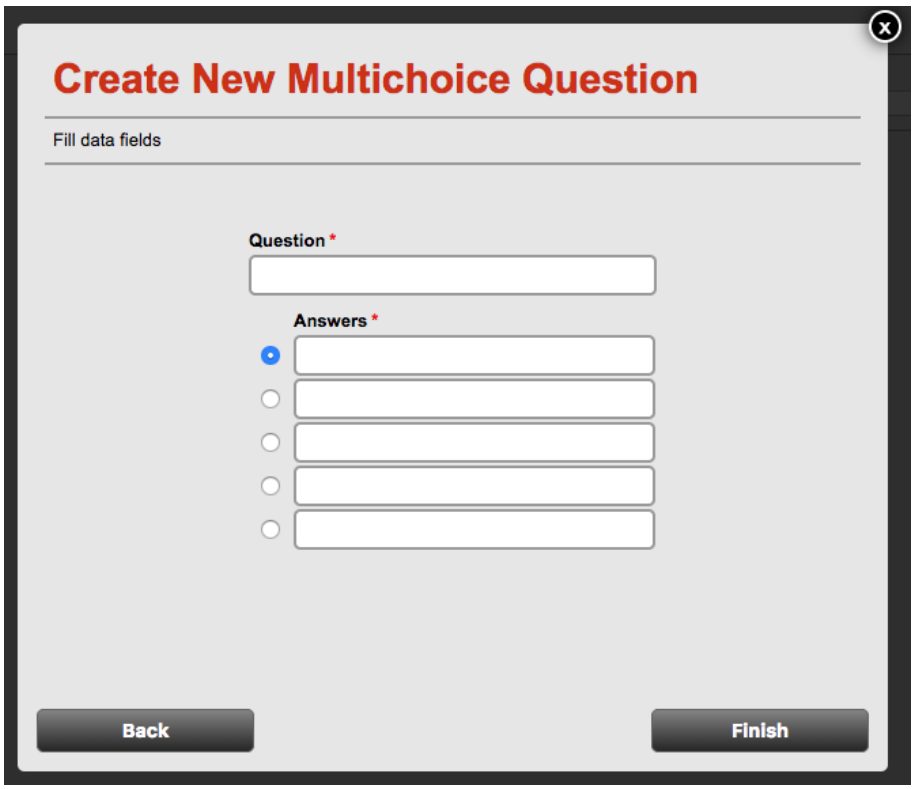


Figura 2.5: Formulário de criação de questões escolha múltipla

que estão interligados entre si.
As atividades e os recursos, assim como os projetos, podem ser criados na *dashboard* Epik, na sua respetiva secção (Figura 2.5), ou na área de trabalho o que permite ao utilizador a versatilidade de criar várias atividades de seguida na *dashboard* ou se preferir ir alternando entre construção de cenários e criação de atividades durante o desenvolvimento do projeto, consoante as suas necessidades. No caso em que o utilizador pretende criar um grupo de questões é necessário que todas as questões que deseje que façam parte do grupo já estejam criadas previamente na plataforma sendo possível, durante o processo de criação de atividades, atribuir-lhes dicas e recursos para serem usados como ajudas durante o jogo.

2.2.1.2 Criação de cenários

Durante o desenvolvimento de um projeto, atualmente, ao criar um cenário a partir da barra de ferramentas da área de trabalho (Figura 2.4) são apresentadas ao utilizador três opções para criar um cenário: cenário de recursos em branco, cenário de atividades em branco ou cenário com base num *template* que já apresenta alguns conteúdos posicionados para o utilizador apenas preencher.
Na área de trabalho, quando o cenário está selecionado, à esquerda, é apresentado um painel de navegação onde são apresentados todos os componentes do cenário como os

textos, atividades, recursos e formas.
À direita está a barra de propriedades onde é possível alterar o nome do cenário, cor de fundo e o fluxo a partir do mesmo, ou seja, para que cenário o jogo avançará em cada uma das quatro opções que existem atualmente. Quando qualquer conteúdo do cenário é selecionado a barra de propriedades apresenta outras opções como: nome, tamanho, posição, texto e cor. No caso do conteúdo selecionado ser uma atividade é possível o utilizador definir suas pontuações, ajudas, bónus e, caso o jogo seja colaborativo, o tempo limite de resposta após receber uma ajuda e em relação aos grupos de questões é possível ainda definir para cada questão, na área de ajuda, quais as ajudas que devem estar disponíveis para o jogador, porém, caso todas as ajudas tenham sido selecionadas mas não se adequem à pergunta esta não será apresentada ao jogador. Por exemplo: se a opção de 50/50 for selecionada e a questão não for de escolha múltipla, não será apresentada ao jogador essa opção.
Estas propriedades referentes às atividades são atribuídas ao cenário porque é possível reutilizar as atividades em vários cenários alterando estas propriedades de modo a enquadra-las no projeto em questão.

2.2.1.3 Geração de jogos

Ao terminar o desenvolvimento do jogo, utilizador tem que exportar o seu jogo através da *dashboard* na secção de projetos ou através da área de trabalho na barra de ferramentas. No processo de exportação o utilizador tem dois campos obrigatórios (nome do jogo e visibilidade) e um campo facultativo (ícone), caso o utilizador não personalize o ícone será utilizado o ícone do Epik. A visibilidade de um jogo é o que define se a informação do jogo e registo de sessões está aberto publicamente, no entanto qualquer jogador com acesso ao URL do jogo poderá aceder ao jogo mesmo que não esteja registado no Epik, sendo esta uma das falhas a colmatar no futuro. Ao aceder ao jogo, o nome e o ícone serão apresentados no cenário inicial do jogo.
Depois de submeter a informação base do jogo, o Epik validará os dados de jogo e caso a validação seja cumprida com sucesso o jogo fica disponível na *dashboard* através da secção de jogos, caso contrário é apresentado ao utilizador uma lista de erros a resolver no projeto. No processo de validação são verificadas as seguintes regras [5]:

- O jogo tem um inicio e um fim - deve ser especificado um cenário como primeiro e deve existir pelo menos um cenário com salto para o cenário de *ranking*;
- O jogo é acíclico - um cenário não pode redirecionar para outro que já tenha sido jogado evitando assim ciclos;
- Não existem cenários isolados - existe pelo menos um caminho que vai desde o cenário inicial a cada um dos restantes cenários;
- Existem pelo menos dois cenários;

- Não existem cenários vazios;
- O jogo deve conter pelo menos uma atividade;
- Não existem componentes vazios;
- Existem pelo menos duas dicas - as atividades com ajuda devem conter sempre pelo menos duas dicas dentro do tópico onde é possível a escolha de apenas uma.

Após validar estas regras o Epik replica todos os dados do projeto, recursos e atividades utilizados de forma a impedir alterações, portanto, o jogo final é uma replica de todos os dados utilizados no projeto armazenados num formato diferente. Para alterar um jogo é necessário voltar ao projeto e repetir o processo de exportação outra vez para verificar que as alterações não tornaram o jogo inválido.

2.2.1.4 Tecnologias usadas no Epik

O Epik está dividido em quatro camadas: a camada de apresentação, a camada de lógica, a camada de dados e a camada de comunicação [5].
Na camada de apresentação foi usado *HyperText Markup Language 5* (HTML5) e *Cascading Style Sheets 3* (CSS3), sendo estas tecnologias recentes, são fáceis de usar e não necessitam outras tecnologias para a criação de elementos gráficos numa página Web. Na camada lógica temos que converter o formato do projeto para conseguirmos enviar para o cliente, para isto são usados mecanismos de conversão de XML para JSON e vice-versa. A camada de dados é composta por duas bases de dados MySQL [14] e os dados dos projetos em XML. Na camada de comunicação é utilizada a biblioteca Socket.IO [15] que nos permite tratar dos pedidos que estão construidos em JSON.
O servidor do Epik foi desenvolvido usando o NodeJS [16] 0.8.16, pois permite o desenvolvimento na mesma linguagem que no cliente, JavaScript.

2.3 Jogos Epik

Os jogos Epik têm como objetivo dar aos jogadores uma forma interativa de aprender quer seja na sua versão *singleplayer* ou na sua versão *multiplayer*, estes jogos são conjuntos de cenários ligados entre si que podem conter recursos e/ou atividades, existe ainda uma pontuação final para motivar os jogadores e na sua versão *multiplayer* do jogo, o Epik apresenta um mecanismo de colaboração, em forma de ajudas para tornar o jogo mais interativo.

2.3.1 Cenários

Na execução de um jogo Epik verifica-se sempre a existência de seis tipos de cenários como representado na Figura 2.6 [5]:

- **Inicio** - o cenário inicial onde é apresentado o titulo do jogo;
- **Instruções** - o cenário onde podemos consultar as regras dos jogos Epik;
- **Sala de espera** - o cenário onde o jogador define o seu avatar e espera por outros colegas em caso de jogo *multiplayer*;
- **Cenários de corpo** - o conjunto de cenários definidos pelo criador do jogo onde serão resolvidas as atividades ou consultados recursos;
- **Fim de jogo** - o cenário final caso o jogo não seja concluído com sucesso;
- **Classificações** - o cenário final onde é apresentada a pontuação final e os *rankings* após conclusão do jogo com sucesso.

Os cenários de corpo de um jogo Epik podem ser [5]:

- *Lecture Scenario* onde o utilizador disponibilizará recursos para o jogador consultar, se preparar para as questões seguintes e rever a matéria das questões anteriores;
- *Activity Scenario* onde serão adicionados ao cenário um conjunto de atividades, que podem conter também um conjunto de ajudas definidas pelo utilizador, para o jogador resolver.

Na Figura 2.6 [5] é possível ver como estes cenários se interligam e no caso dos cenários de corpo, atualmente na plataforma existem quatro mecanismos de controlo no fluxo de cenários, o que torna a evolução do jogo limitada. A passagem de um cenário para outro, nos jogos *singleplayer* ou *multiplayer*, só acontece na presença de uma das quatro situações seguintes:

- O tempo disponível para realização das atividades no cenário chega ao fim;
- Todos os jogadores concluem as atividades do cenário;
- Todos os jogadores fazem *Skip* do cenário;
- Todos os jogadores pressionam o botão ”Continuar”.

O facto de existir evolução apenas na presença destas situações torna o fluxo muito limitado obrigando, no caso do jogo colaborativo, a que todos os jogadores se encontrem no mesmo cenário, o que penaliza os jogadores mais rápidos metendo o jogo em pausa à espera que os seus colegas acabem o cenário anterior. As transições existentes não têm em conta o desempenho dos jogadores uma vez que todos prosseguem ao longo do fluxo de cenários sempre em simultâneo. Para eliminar esta limitação serão criados outros mecanismos de evolução no fluxo de cenários e assim como a possibilidade dos jogadores estarem em cenários diferentes, dependendo de seu nível de conhecimento no jogo.

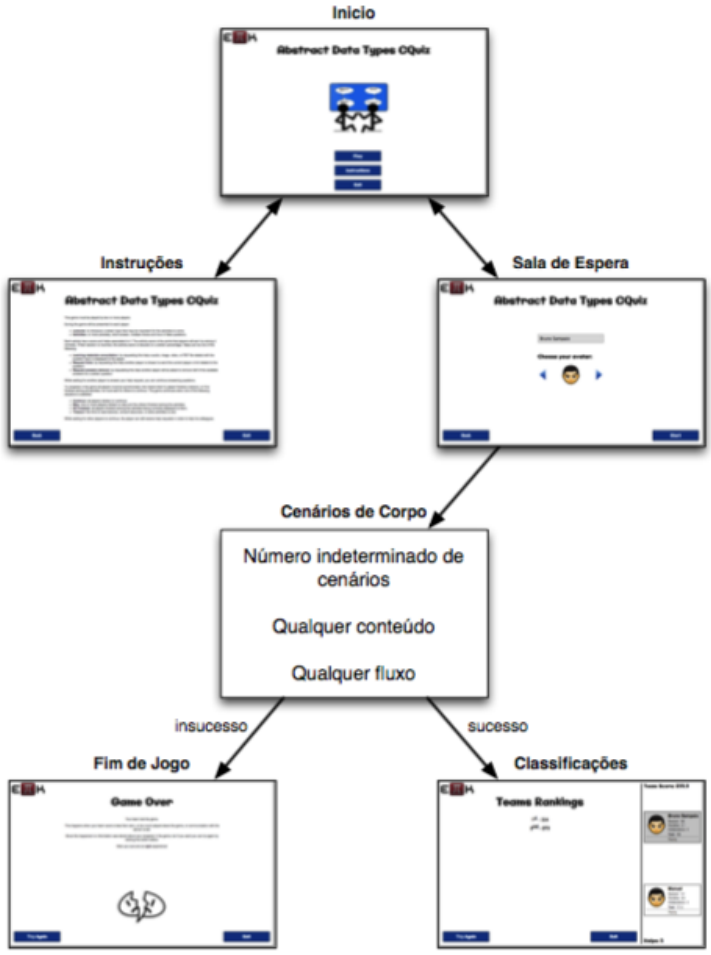


Figura 2.6: Fluxo de Cenários

2.3.2 Atividades e recursos

Os recursos usados nos jogos Epik podem ser imagens, vídeos, PDF ou áudio, são apresentados nos cenários por um *sprite* e apenas são visualizados após o clique do jogador, com exceção das imagens.
Por sua vez, as atividades nos jogos Epik são questões de escolha múltipla, verdadeiro ou falso e resposta curta, que para além de uma atividade individual, também se pode inserir num cenário de um jogo um grupo de questões onde desse grupo será escolhida uma questão aleatoriamente para cada jogador no momento de execução do jogo.
Em cada atividade pode existir um número limitado de ajudas e não é possível o jogador ganhar mais ajudas ao longo do jogo, tendo estas ajudas influencia na pontuação final dos jogadores, penalizando ao pedir ajudas e recompensando aqueles que ajudam. As ajudas associadas a cada atividade podem ser de três tipos [5]:

- **50/50** - Disponível apenas para as perguntas de escolha múltipla onde são retiradas metade das respostas disponíveis, nunca ocultando a resposta correta;

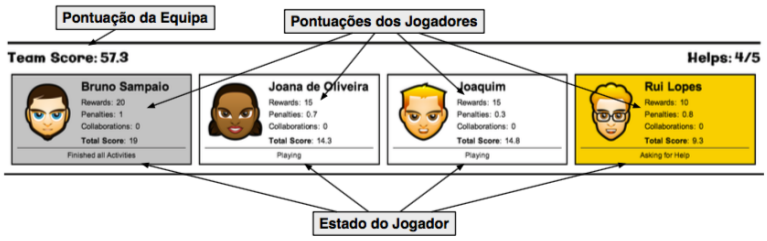


Figura 2.7: Painel de jogadores do jogo Epik

- **Dicas** - É apresentada ao jogador uma frase curta associada à atividade, que ajuda na resolução;
- **Consulta de Recursos** - O jogador tem acesso a um recurso relativo ao tópico da atividade em forma de PDF, imagem, áudio ou vídeo.

Atualmente na plataforma apenas é possível criar perguntas de escolha múltipla, resposta curta ou verdadeiro ou falso o que pode tornar o jogo monótono e pouco dinâmico.

2.3.3 Painel de jogadores

Aquando da execução dos jogos Epik existe um painel de jogadores (Figura 2.7) [5] onde é possível em cada momento ver o avatar, nome, estado e as respetivas pontuações de cada jogador para que desta forma a competitividade entre os jogadores aumente e motivando para um melhor desempenho. Cada jogador está num de cinco possíveis estados:

- A jogar;
- Fez *Skip* do cenário;
- Clicou em Continuar;
- Terminou todas as atividade;
- Atingiu o tempo limite.

E as suas pontuações estão divididas em:

- Pontos de recompensa - pontos recebidos por cada atividade, terminar primeiro ou por colaboração bem sucedida;
- Pontos perdidos - penalização por cada resposta errada ou ajuda sem sucesso;
- Total de pontos - a diferença entre pontos de recompensa e pontos perdidos;

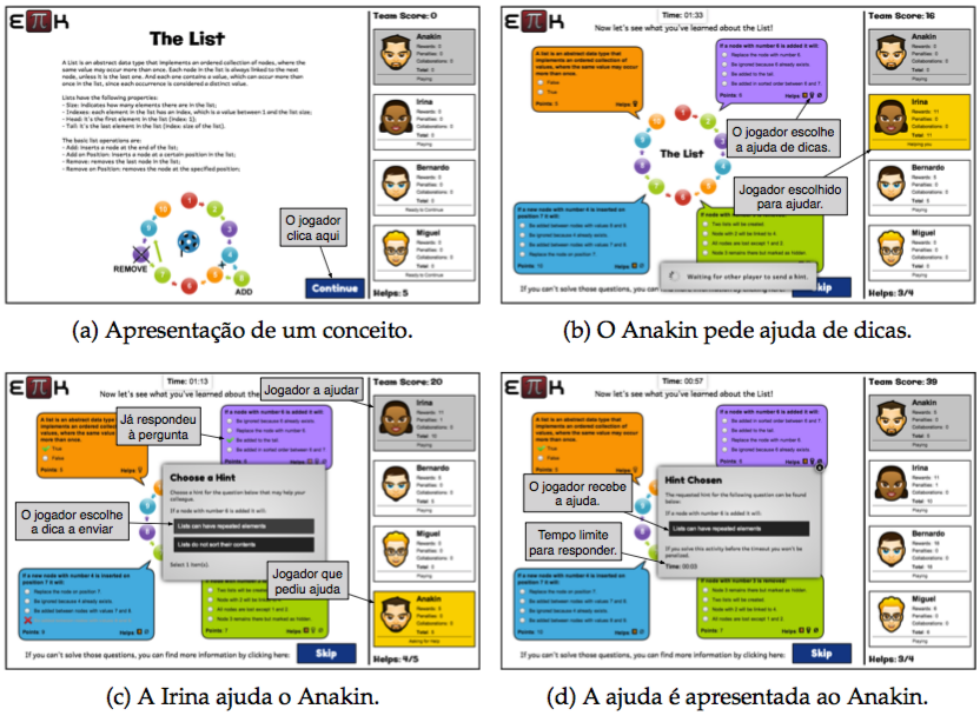
O painel de jogadores atualmente implementado disponibiliza também o número total de ajudas do jogador e as ajudas que ainda podem ser usadas no cenário atual no formato k/n onde n é o total de ajudas e k as que ainda estão disponíveis.

2.3.4 Colaboração

Os jogos Epik divergem dos outros pelo fato de permitir colaboração, contudo os jogos colaborativos necessitam de regras especiais como por exemplo, os tempos de respostas, os bónus e a pontuação da equipa.
Quando estamos perante um jogo Epik colaborativo é necessário entre dois a quatro jogadores para executar este tipo de jogo sendo também imprescindível que estes jogadores estejam sempre todos no mesmo cenário, pois o jogo só prossegue quando todos os jogadores estiverem prontos a avançar. Na execução de um jogo colaborativo existem dois novos estados de jogador: a ajudar e a pedir ajuda.
Para ganhar pontos de colaboração os jogadores precisam de receber um pedido de ajuda para a pergunta que acabou de responder com sucesso. Por sua vez o jogador deve retirar respostas erradas da questão ou ajudar com uma das dicas associadas à atividade, algo que o sistema faz no modo *singleplayer* do jogo. É ainda apresentado um tempo limite para que a pergunta seja respondida e determinar se o bónus é aplicado ou não, para tal o jogador que pediu ajuda deve resolver a questão dentro do tempo limite e sem nenhuma tentativa falhada.
Na Figura 2.8 [17] é apresentado um exemplo de um pedido de ajuda onde o Anakin pede ajuda e o sistema seleciona um dos outros jogadores que já respondeu corretamente à atividade, neste caso a Irina. A Irina seleciona a dica que acha mais adequada para ajudar o Anakin de uma lista associada à atividade anteriormente. Por fim, a dica é apresentada ao Anakin que terá um tempo limite para responder corretamente sem ser penalizado e dando um bónus à Irina.

2.4 Sumário

Neste capitulo é realizado trabalho de pesquisa e introdução teórica aos jogos como ferramentas educacionais e como a plataforma Epik abrange estes conceitos. Podemos concluir que existem melhoramentos a fazer na plataforma, sendo um deles a necessidade de existirem fluxos independentes na execução de um jogo Epik consoante o conhecimento dos jogadores.



(a) Apresentação de um conceito.

(b) O Anakin pede ajuda de dicas.

(c) A Irina ajuda o Anakin.

(d) A ajuda é apresentada ao Anakin.

Figura 2.8: Exemplo de colaboração no Epik

Capítulo 3

Proposta de solução

De modo a resolver algumas fragilidades anteriormente descritas do Epik e incorporar novas funcionalidades foi desenvolvida uma nova plataforma com novos jogos, sendo esta dividida em duas áreas:

- *Aplicação desktop - onde o utilizador poderá criar projetos definindo os seus cenários constituídos por atividades e recursos, criar atividades, importar recursos e exportar jogos;*
- *Servidor de execução de jogos - alojado juntamente com o site do Epik, onde os utilizadores poderão registar-se para importar os seus jogos e os poder partilhar com jogadores ou utilizadores.*

Com esta constituição vai ser possível aos utilizadores desenvolver os seus jogos sem a necessidade de ligação à Internet e ter os seus ficheiros guardados localmente antes de os importar no servidor de execução.

Uma das vantagens de tornar o ambiente de desenvolvimento do Epik numa aplicação *desktop* é a possibilidade de criar os seus projetos sem a necessidade de estar conectado à internet, apenas o será necessário no momento de download da aplicação no site do Epik ou quando o utilizador terminar o projeto e o quiser importar para o servidor de execução de jogos onde ficará disponível para ser partilhado através do endereço de email.
Para realizar esta transformação é necessário uma discussão sobre qual a melhor *framework* a usar permitindo exportação da aplicação para múltiplas plataformas, utilizando linguagens às quais o grupo seja familiar e com compatibilidade para com as ferramentas necessárias ao desenvolvimento das tarefas individuais de cada elemento do grupo de trabalho.

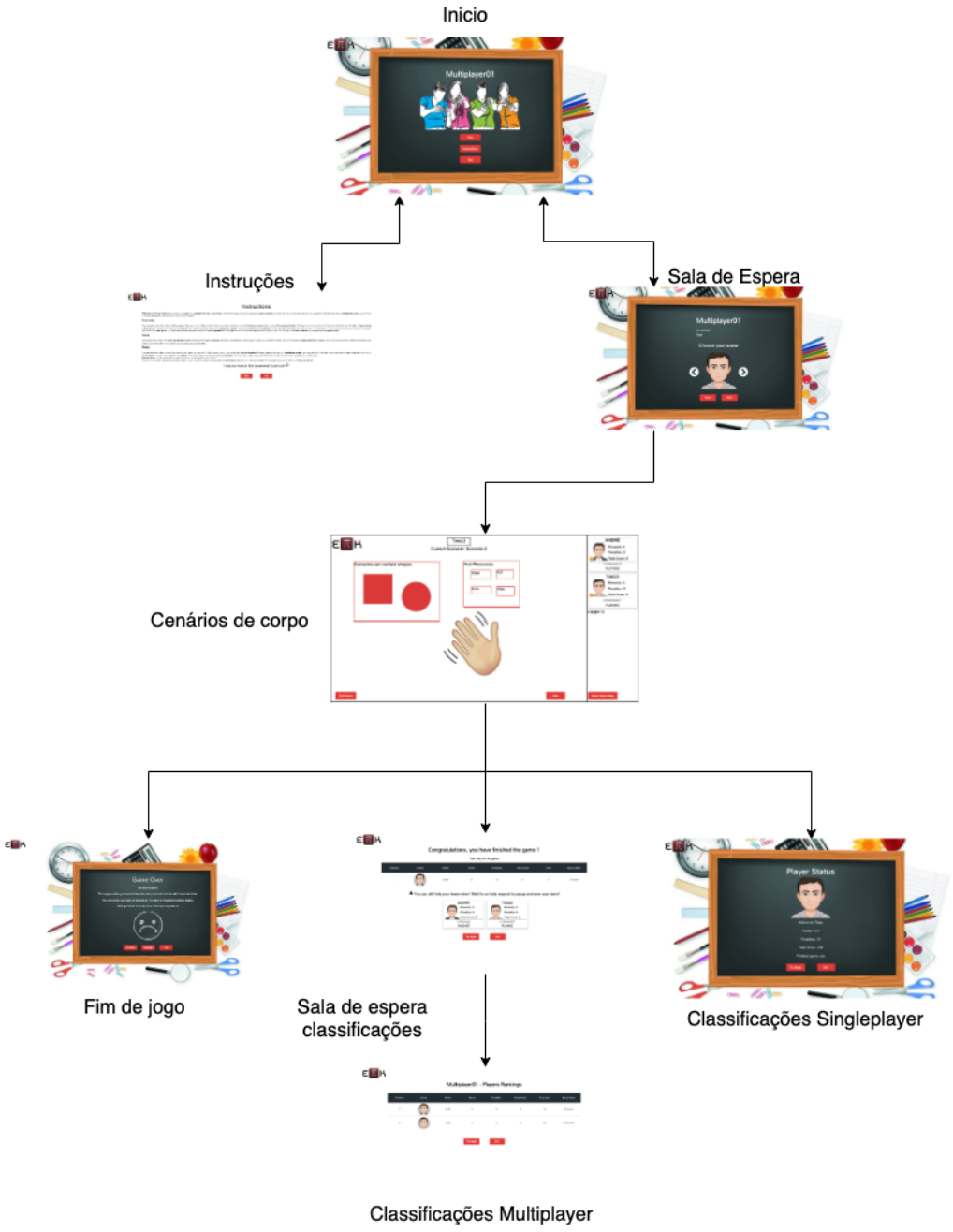


Figura 3.1: Fluxo de Cenários Epik atualizado

3.1 Novos jogos Epik

Como referido na secção 2.3, os jogos Epik têm como objetivo dar aos jogadores uma forma interativa de aprender quer seja na sua versão *singleplayer* ou na sua versão *multiplayer*. Estes jogos são conjuntos de cenários ligados entre si que podem conter recursos e/ou atividades, existindo ainda uma pontuação final para motivar os jogadores e verifica-se sempre a existência de seis tipos de cenários como representado na Figura 3.1:

- **Inicio** - o cenário inicial onde é apresentado o titulo do jogo;

- **Instruções** - o cenário onde podemos consultar as regras dos jogos Epik;
- **Sala de espera** - o cenário onde o jogador define o seu avatar e espera por outros colegas em caso de jogo *multiplayer*;
- **Cenários de corpo** - o conjunto de cenários definidos pelo criador do jogo onde serão resolvidas as atividades ou consultados recursos;
- **Fim de jogo** - o cenário final caso o jogo não seja concluído com sucesso;
- **Sala de espera classificações** - cenário em que, caso o jogo seja *multiplayer*, os jogadores esperam pelo resultado final antes de aceder às classificações;
- **Classificações** - o cenário final onde é apresentada a pontuação final e, em caso de jogo *multiplayer*, os *rankings* da sessão após conclusão do jogo com sucesso.

Antigamente no Epik, na sua versão anterior, os jogos eram constituídos por cenários compostos por atividades de questões e recursos e caso o jogo fosse *multiplayer* os jogadores para evoluir no jogo precisavam de estar todos no mesmo cenário não permitindo assim uma grande variedade e flexibilidade na construção dos fluxos de jogos. Esta limitação da versão *multiplayer* do jogo foi eliminada de modo a permitir um jogo mais dinâmico dando a possibilidade aos jogadores de seguirem o jogo independentemente sem a necessidade de estarem todos os jogadores no mesmo cenário.
As atividades já existentes foram reestruturadas e foram adicionados dois novos tipos, agora, nesta versão, é possível também inserir atividades de puzzle e atividades de programação aumentando assim a variedade de jogos e as possibilidades de criação de cenários para o utilizador. Porém, as atividades não vêm instaladas na aplicação base do Epik, caso o utilizador deseje um determinado tipo de atividade terá que efetuar o *download* do respetivo *plugin* no *website* da plataforma.

3.1.1 Novas transições de cenário

De modo a melhorar os pré requisitos do fluxo de jogos descritos anteriormente, obter objetivos alcançáveis e adequados a cada jogador foram implementadas novas transições de cenário nos jogos Epik.
Com a nova versão do Epik, a base das transições foi repetida para os cenários gerais como referido na Figura 2.6. No entanto, nos cenários de corpo tendo como por objetivo retirar a linearidade de um jogo Epik e adequar o jogo a cada jogador foram adicionadas novas transições de cenários às três já existentes. Os três tipos de transição existentes na versão anterior do Epik e que foram mantidas nesta nova implementação são:

- ***Skip*** - em que o jogador clica num botão que permite a passagem para um cenário seguinte;

- ***Timeout*** - o jogador deixa que o tempo definido pelo utilizador para o cenário se esgote;
- ***Finish*** - as atividades incluídas no cenário são dadas por terminadas. Em que este tipo de transição só é apresentado ao utilizador caso exista um *plugin* de atividades instalado na aplicação.

De modo a adaptar o jogo a cada jogador e tornar o mesmo mais interativo é agora dada ao utilizador, se na instalação atual da plataforma já existir algum *plugin* de atividades instalado, a possibilidade de escolha entre a utilização ou não de conhecimento como base das transições do fluxo de cada cenário. Ao selecionar o modo de transição com conhecimento é possível ainda o utilizador decidir se quer utilizar duas divisões de conhecimento e separar apenas os jogadores acima ou abaixo dos 50% ou se quer premiar a excelência e utilizar três divisões de conhecimento, dos 0% aos 50%, dos 50% aos 90% e dos 90% aos 100%. Para além de adicionar as novas transições, uma das mudanças mais relevantes nos jogos *multiplayer* para esta nova versão foi a possibilidade dos jogadores avançarem no jogo Epik ao seu ritmo, sem ter que esperar que todo o grupo termine o cenário para poder avançar no jogo.
Assim sendo, no novo Epik para além das três antigas transições é também possível adicionar divisões de conhecimento na base qualquer um desses tipos de transição:

- Duas divisões de conhecimento:
  - Dos 0% aos 50% da pontuação completa do cenário.
  - Dos 50% aos 100% da pontuação completa do cenário.
- Três divisões de conhecimento:
  - Dos 0% aos 50% da pontuação completa do cenário.
  - Dos 50% aos 90% da pontuação completa do cenário.
  - Dos 90% aos 100% da pontuação completa do cenário.

Com este novo tipo de transição é possível o utilizador adaptar o jogo a cada jogador, definindo fluxos de cenários com vários níveis de dificuldade e permitindo que cada um faça o seu percurso de forma adequada ao seu desempenho e de acordo com os cenários definidos pelo utilizador.

3.1.2 Novo *feedback* para jogadores

O fluxo de execução nos jogos é essencial para manter o jogador focado e motivado na tarefa que está a realizar e, no caso de jogos educacionais, este fluxo torna-se ainda mais importante porque o jogador necessita de ter sucesso para manter a motivação, no entanto se o sucesso for facilmente atingido a motivação é substituída por monotonia e o jogador deixa de jogar. Caso o jogo proposto não seja apresentado com clareza sobre o que

fazer e como fazer para atingir o sucesso ou seja de exagerada dificuldade de resolução, é normal o jogador sentir frustração e ansiedade, algo que fará com que também deixe de jogar. Posto isto, um dos componentes fundamentais no fluxo de execução de jogos é o *feedback*, que vai permitir aos jogadores saber imediatamente os resultados das suas ações. Anteriormente como todos os jogadores tinham que estar no mesmo cenário o jogador apenas recebia *feedback* sobre pontuações e sobre o estado de cada jogador: A jogar, Fez *Skip* do cenário, Clicou em Continuar, Terminou todas as atividade ou Atingiu o tempo limite.

Agora, na nova versão como cada jogador é independente, todos estarão a jogar e o *feedback* apresentado será na área do mapa do jogo, onde é possível ter uma vista geral do jogo, ver em que cenário se encontra, o que necessita de fazer para chegar ao fim e no caso de ser um jogo *multiplayer*, uma tabela onde são apresentadas as pontuações e o cenário onde cada jogador se encontra. Desta maneira é possível ao jogador saber e comparar o seu desempenho com os restantes colegas, despertando assim o espírito de competitividade.

3.2 Desenvolvimento de jogos

A plataforma Epik tem como por objetivo permitir o desenvolvimento de jogos aos utilizadores mesmo sem que estes tenham algum conhecimento de programação, então para atingir este objetivo, a aplicação *desktop* é composta por uma interface gráfica que permite a criação de cenários sem necessidade de conhecimentos de programação.

3.2.1 Nova plataforma Epik

De modo a permitir o desenvolvimento de jogos sem a necessidade de ligação à Internet, a nova plataforma Epik, será implementada com base numa *framework* própria para desenvolvimento de aplicações *desktop*. A aplicação está disponível sem o desenvolvimento de atividades no site do Epik para as plataformas MacOS e Windows. Caso o utilizador deseje inserir atividades, estas são *plugins* independentes de forma a não sobrecarregar com ficheiros indesejados a plataforma e também estão disponíveis em separado na página web da aplicação.
Após efetuar o download e abrindo a plataforma pela primeira vez é disponibilizado ao utilizador o manual de utilização em cada uma das áreas para que a integração e utilização da plataforma seja fácil sem ter que pesquisar fora da aplicação sobre como desenvolver o seu jogo. Este manual pode ser desativado a qualquer momento para que os utilizadores mais experientes possam utilizar a plataforma sem ter que passar pelo tutorial representado na Figura 3.3. A nossa plataforma é composta por uma *dashboard* e por uma área de desenho do jogo. Na *dashboard* temos a página inicial (Figura 3.2) que

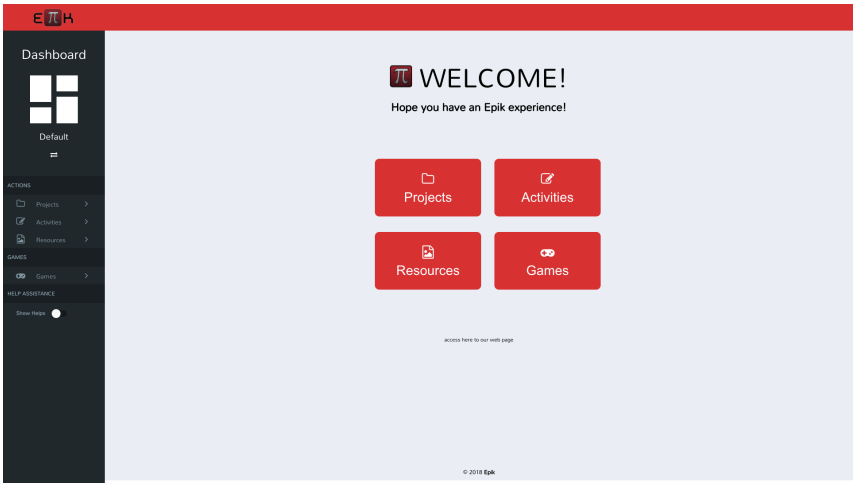


Figura 3.2: Página inicial Epik

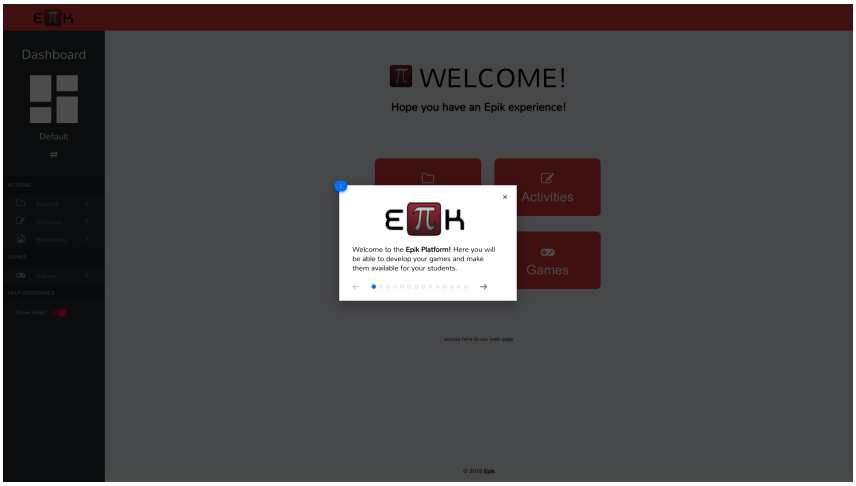


Figura 3.3: Manual integrado do Epik

nos redireciona para as listagens dos projetos, das atividades e dos recursos importados para a plataforma (Figura 3.4).

Em todas as três secções é possível editar a informação base (como nome e descrição), apagar o elemento selecionado ou adicionar um novo projeto, atividade ou recurso à plataforma como na Figura 3.5. Caso estejamos perante a secção das atividades, antes de ser possível criar e editar é necessário importar um *plugin* para adicionar cada um dos tipos de atividades à aplicação. Para além desta funcionalidade, as transições com base no conhecimento dos jogadores só ficarão disponíveis para implementação no projeto caso exista pelo menos um *plugin* instalado. No caso dos projetos existe também a opção de editar o modo de jogo escolhendo *singleplayer* ou *multiplayer* e de exportar o projeto como jogo caso este projeto já tenha sido validado de acordo com as regras dos jogos Epik. Ao selecionar o projeto em que deseja trabalhar, o utilizador entra na área de desenho de jogo Epik. Esta área está dividida em três secções:

- Barra de propriedades (Figuras 3.6, 3.7, 3.8) - localizada no topo da página, serve

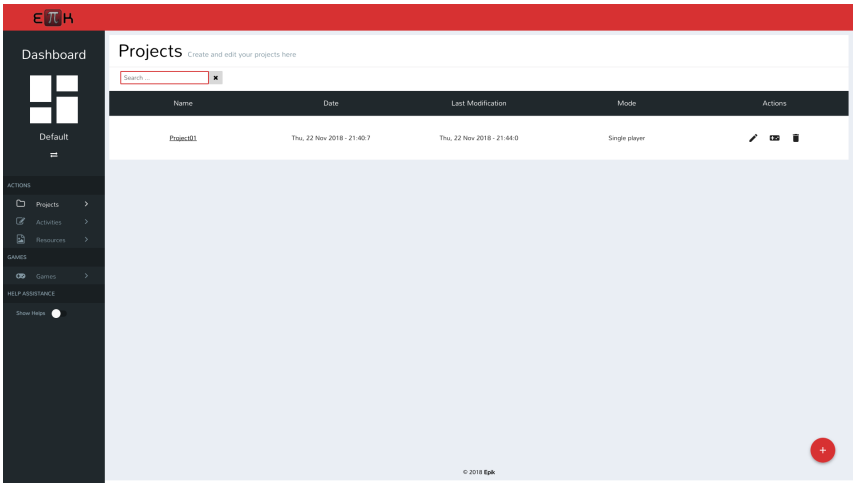


Figura 3.4: *Dashboard* Epik

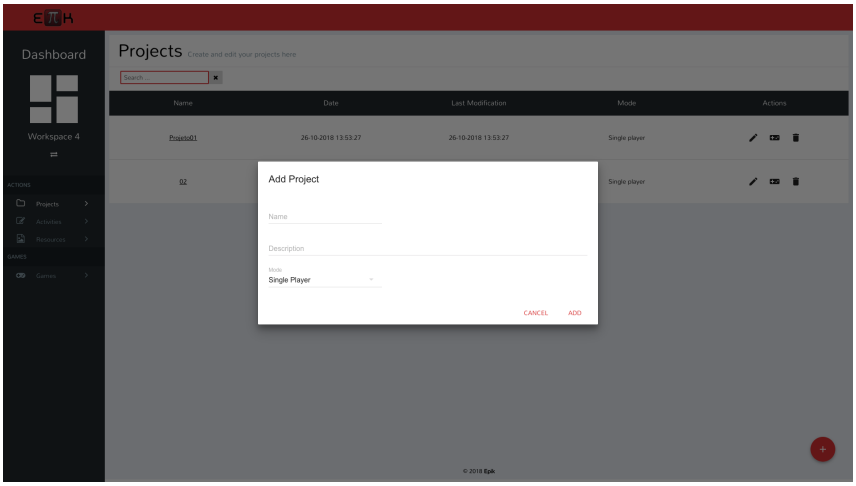


Figura 3.5: Adicionar projeto à plataforma Epik

para organizar e definir todas as propriedades do elemento selecionado, quer seja um recurso, uma atividade, um cenário ou uma das propriedades gerais;

- Explorador de cenários (Figura 3.9) - à esquerda da página, permite navegar entre cenários, recursos e atividades. Existe no explorador também a funcionalidade de adicionar novos cenários ao projeto;
- Área de edição (Figura 3.10) - onde o utilizador pode compor com recursos e atividades o cenário selecionado previamente no explorador.

A barra de propriedades é atualizada consoante as propriedades que o utilizador necessitar para o elemento selecionado no momento, sendo os quatro grupos de propriedades mais comuns, as propriedades do projeto, do cenário, do recurso e das atividades. Nas propriedades do projeto, representadas na Figura 3.6, estão as funcionalidades de criar um novo projeto, guardar e exportar o projeto em formato de jogo. É possível aceder às opções gerais do projeto como a posição do logo, propriedades da barra de estado do

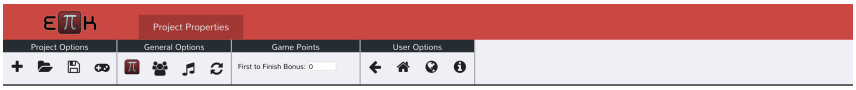


Figura 3.6: Barra de propriedades do projeto

Figura 3.7: Barra de propriedades do cenário

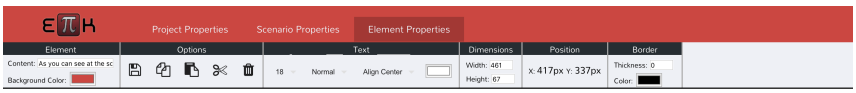


Figura 3.8: Barra de propriedades de recursos

jogo, áudios e colaboração existente no projeto através do primeiro separador existente no explorador de cenários.

As propriedades do cenário, representadas na Figura 3.7, permitem ao utilizador a criação de novos cenários, guardar o projeto e editar a cor para o fundo, o número de ajudas, tempo disponível para o jogador no cenário e definir o cenário como inicial. Nesta barra de propriedades foi implementada a funcionalidade que permite aos utilizadores adicionarem recursos (texto, imagem, vídeo, áudio e PDF) e figuras (quadrados, círculos e balões), assim como copiar, cortar, colar e eliminar o elemento selecionado do cenário. Caso o utilizador decida criar todas as transições todas as transições das quais o cenário selecionado é a origem é possível através do penúltimo ícone da barra de propriedades de cenários. Ao clicar no ícone é apresentado um modal que indica todos os tipos de transição possíveis para o cenário e o utilizador através de menus *dropdown* escolhe o cenário de destino para determinada transição.

A barra de propriedades dos recursos, na Figura 3.8, admite a edição do estilo de um recurso previamente selecionado. Apresenta a sua posição no cenário, permite a edição de conteúdo de texto, tamanho de letra, tipo, alinhamento e cor, dimensões, cor de fundo, rebordo, cor do perímetro do elemento e implementa também as funcionalidades de guardar o projeto, copiar, cortar, colar e eliminar o elemento selecionado.

Do lado esquerdo do componente está o explorador de cenários, na Figura 3.9. Esta barra lateral permite-nos adicionar, editar o nome ou eliminar cenários, navegar entre elementos como cenários, atividades e recursos apresentando-os selecionado na área de desenho do jogo. Para além destas funcionalidades referidas anteriormente é também possível editar as propriedades do logo, dos jogadores, sons e colaboração no jogo abrindo um novo separador correspondente na barra de propriedades.

Na área de desenho, apresentada na Figura 3.10, é onde o utilizador compõe, adicionando figuras, recursos e/ou atividades, e pré-visualiza a estrutura do seu cenário através da interface gráfica que lhe é proporcionada, sendo nesta área possível mover e redimensionar os componentes livremente dentro dos limites da área. Para além da edição através de

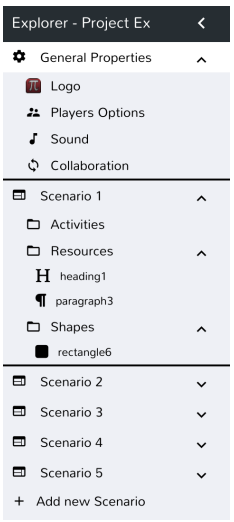


Figura 3.9: Explorador de cenários

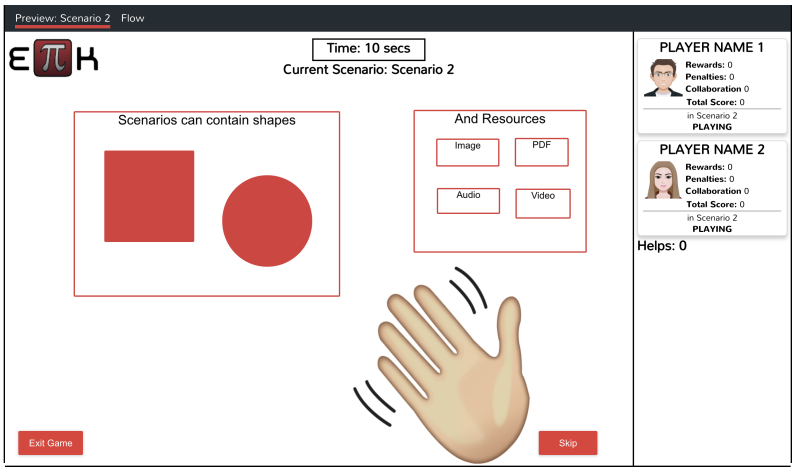


Figura 3.10: Área de desenho

mecanismos de *drag-and-drop* também é possível editar recursos através da barra de propriedades do elemento em que esta é atualizada segundo o ultimo elemento selecionado apresentando as propriedades atuais do mesmo e atualizando ao *input* do utilizador.

3.2.2 Validação e geração de jogo Epik

Os jogos Epik para serem adicionados ao servidor de execução primeiro tem que ser exportados do ambiente de desenvolvimento gerando um ficheiro .ZIP composto pelo ficheiro de jogo no formato .JSON e todos os recursos que estão presentes no projeto.
Para isso, na *dashboard* do ambiente de desenvolvimento e na barra de propriedades do projeto existe a opção de exportar o jogo. Esta opção não só exporta o jogo Epik como também verifica a validade do mesmo, que foi testada durante a funcionalidade de guardar projeto. No momento em que o projeto é guardado, caso o jogo não seja válido, um alerta é emitido pela aplicação consoante a validação que está a falhar, caso contrário

a base de dados da aplicação é atualizada com a informação de que este projeto é válido e está pronto a ser exportado. O teste para validação do projeto é composto pela seguinte lista de verificações:

- Existe um cenário de inicio;
- Existe um cenário com transição para o cenário de classificações;
- Não existe ciclos no fluxo de cenários;
- Todos os cenários são alcançáveis a partir do primeiro cenário;
- A partir de todos os cenários é possível chegar ao cenário de classificações;
- Existe mais do que um cenário no jogo;
- Não existem cenários vazios;
- No caso de existir fluxo com base no conhecimento verifica que todas ou nenhuma das divisões de cada tipo de transição estão atribuídas.

Ao exportar, após confirmar na base de dados que o jogo é válido, é pedido ao utilizador para escolher onde deseja guardar o ficheiro .ZIP e é gerado um ficheiro .JSON dentro do .ZIP em que o nome é composto pelo identificador do projeto e a data em que está a ser criado o ficheiro. O ficheiro é composto por duas chaves principais: a informação básica do projeto que é um objeto com os valores do nome, identificador e tipo de jogo, e um outro objeto com vetores para os cenários e para os recursos de cada cenário que compõem o projeto. Em que este vetor é composto por cenários compostos por um objeto com a informação do cenário, por um vetor de recursos e por um vetor de atividades correspondentes a esse cenário. No cenário para além do identificador, nome e descrição cada cenário tem também um vetor com a informação das suas transições (utilizando para definir o identificador do cenário seguinte), se é cenário inicial, se e como usa o conhecimento na baseadas das transições. No vetor de recursos cada objeto tem disponível o seu identificador, nome, descrição e todos os parâmetros de estilo necessários para disponibilizar os recursos no servidor de desenvolvimento, como por exemplo a cor, o tamanho, o rebordo, a posição no cenário e o tipo e tamanho de letra caso seja necessário exibir texto. No vetor de atividades para além das pontuações e posição no cenário está também disponível o tipo e o subtipo da atividade em questão e consoante estes valores são apresentados os atributos necessários para cada tipo de atividade.

3.3 Componente gráfica para edição de fluxos

Os jogos Epik, devido às suas características de fluxo, podem ser representados como grafos orientados acíclicos. Para esta representação no cabeçalho da área de desenho é apresentado o separador de *flow*, que ao clicar no mesmo, a área de desenho é substituído

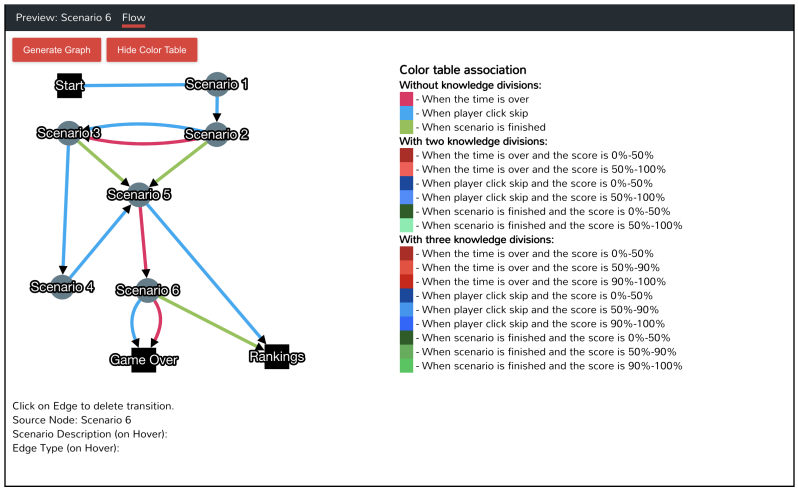


Figura 3.11: Componente gráfica para edição de fluxos

pela componente gráfica para edição de fluxos apresentada na Figura 3.11, esta componente é composta pelo grafo, a tabela de cores e a tabela de informações. De modo a não sobrecarregar a aplicação, o grafo apenas é gerado ao clique do botão "*Generate Graph*"e o botão "*Show Color Table*"é utilizado para apresentar ou ocultar a legenda de cores das arestas do grafo.

Para implementação desta área é necessário uma ferramenta que seja integrável com a *framework* utilizada na aplicação e permita ao utilizador interagir com o grafo criado. Para melhorar a experiência do utilizador foram implementadas novas funcionalidades, descritas nas seguintes subsecções, de acordo com os objetivos da plataforma Epik e capacidades da *framework* a utilizar:

- Criação de uma transição após clique de dois nós, sendo o primeiro clique no nó de origem;
- Eliminação de uma transição após clique na aresta correspondente;
- Visualização do tipo de transição ao cobrir uma aresta com o ponteiro do rato;
- Visualização da descrição do cenário ao cobrir um nó com o ponteiro do rato;

Para complementar esta interface e dar mais possibilidades ao utilizador, também, é possível definir o fluxo através da barra de propriedades do cenário abrindo o modal de *Game Flow*, demonstrado na Figura 3.12, onde aparecerá a lista de transições e para cada uma um *dropdown* com todos os cenários do projeto para selecionar o nó de destino na respetiva transição.

3.3.1 Criação de uma única transição

O utilizador ao entrar na componente gráfica para criação de fluxos de cenários pode não querer definir logo todas as transições para um cenário no modal da Figura 3.12 e para

Figura 3.12: Modal do *Game Flow* para definir todas as transições de um cenário

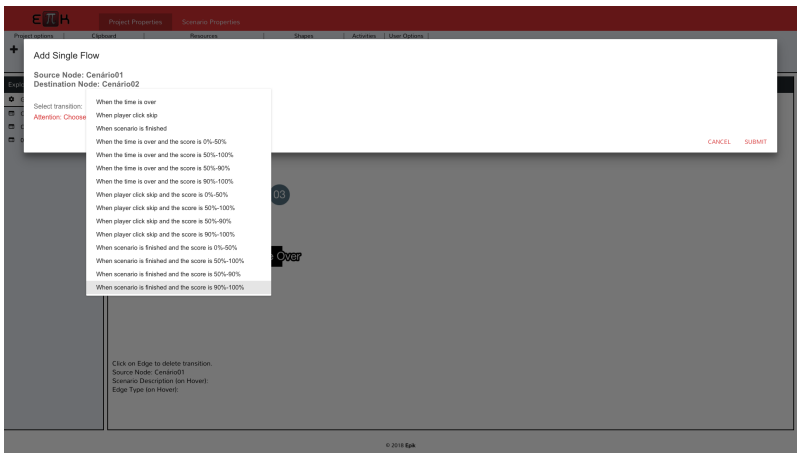


Figura 3.13: Seleção da transição

isso foi desenvolvida esta funcionalidade que permite a criação de uma única transição sem que seja necessário sair da interface gráfica. Para a criação de uma única transição o utilizador, já com o grafo gerado, apenas necessita de selecionar o nó de origem, este aparecerá na tabela de informações do grafo e com uma cor diferente assinalando o destaque, selecionar o nó de destino e automaticamente se abrirá o modal da Figura 3.13 para que o utilizador defina a transição desejada. Ao selecionar a transição vários avisos poderão aparecer avisando do número de divisões de conhecimento que passarão a ser utilizadas, caso este tipo de transição já esteja a ser utilizada, um aviso que a transição será reescrita ou caso a transição seja dependente do tempo que só será adicionada quando o utilizador definir o tempo para realização do cenário. Caso a transição escolhida seja com conhecimento entre os 0% e os 50% é necessário também selecionar o número de divisões utilizadas no fluxo com conhecimento. Por fim, ao submeter a transição desejada o grafo é atualizado automaticamente com a nova transição.

3.3.2 Apagar transição

Para além da opção de reescrever a transição descrita acima é também possível, na componente gráfica, apagar uma transição sem que seja necessário aceder a outros menus. Para

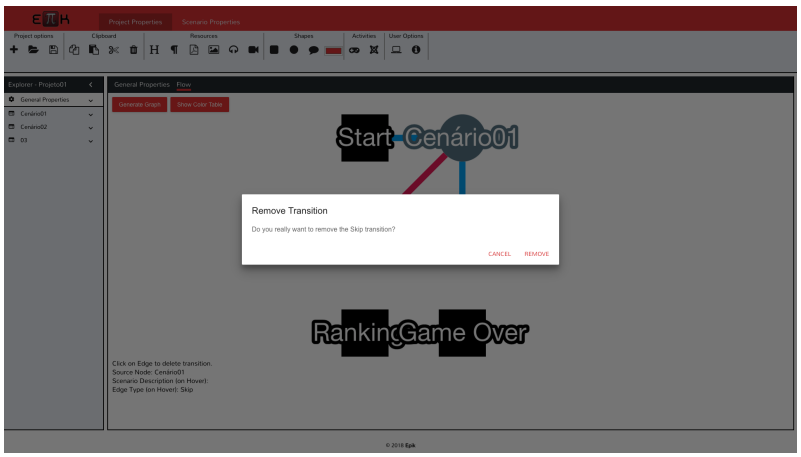


Figura 3.14: Seleção da transição a apagar

Click on Edge to delete transition.
Source Node: Scenario 6
Scenario Description (on Hover): Last scenario of this game. Good luck!
Edge Type (on Hover):

Figura 3.15: Visualização da descrição do cenário ao cobrir um nó com o ponteiro do rato

isto, o utilizador necessita de selecionar a transição que deseja apagar e após confirmar a sua intenção de eliminar a transição no modal da Figura 3.14 o grafo será automaticamente atualizado removendo a aresta selecionada.

3.3.3 Visualização do tipo de transição e descrição do cenário

De forma a facilitar e legendar o grafo foi implementada, por baixo da área do grafo, uma tabela de informações de modo a complementar e tornar o grafo mais intuitivo. Nesta tabela podemos visualizar o nó que está selecionado como origem, a descrição do cenário e o tipo de transição no momento em que o ponteiro do rato está a cobrir o nó, como demonstrado na Figura 3.15, ou aresta correspondente.

3.4 Servidor de execução de jogos Epik

Após a criação do jogo, este pode ser jogado no servidor de execução do Epik. Este servidor está alojado na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa e para ser acedido é necessário estar na rede local ou através de VPN com as credenciais da Faculdade.

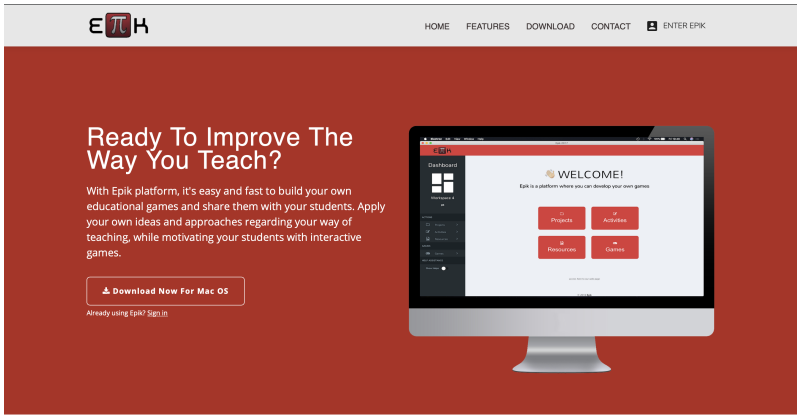


Figura 3.16: Página inicial Epik

3.4.1 Página inicial

Ao aceder ao servidor, antes de fazer *login* na plataforma, é apresentada a página inicial do Epik como representada na Figura 3.16. Nesta página são apresentados os links para download da aplicação *desktop* para MacOS e Windows, as principais características e funcionalidades da plataforma, como o *design* moderno, os vários tipos de atividades, a colaboração como mecanismo para a aprendizagem e o tutorial incorporado na plataforma.
No topo da página temos links que redirecionam o utilizador para cada uma das secções descritas acima.

3.4.2 Área de utilizador Epik

Ao clicar no botão *Enter Epik*, na barra de separadores na parte superior da página, o utilizador terá a opção de se registar na plataforma como criador de jogo, ou de efetuar *login* como criador ou jogador sendo que um criador de jogos pode também entrar na área de jogador.

O utilizador ao decidir que é um jogador, apenas é necessário que insira o seu email na plataforma (Figura 3.18) para ter acesso à sua *dashboard* (Figura 3.19). Nesta página, como é apenas um jogador, o utilizador só terá à sua disposição os jogos que consigo foram partilhados e ao decidir o jogo que quer jogar, será necessário inserir a palavra-passe do jogo como apresentado na Figura 3.20, definida pelo criador, em que após verificação é apresentado o cenário inicial do jogo pronto a jogar.

Caso o utilizador ainda não seja portador de uma conta registada e deseje importar os seu jogos criados, deverá clicar no link a vermelho e preencher o formulário de registo apresentado na Figura 3.21 com o seu nome, email, *password* desejada e concordar com os termos e condições da aplicação. Ao submeter o formulário, um novo utilizador é criado na base de dados e é redirecionado para a *dashboard* do servidor onde poderá importar e

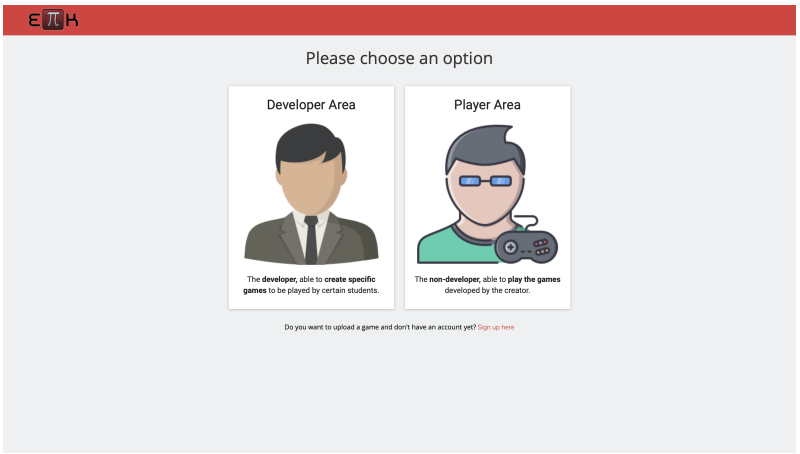


Figura 3.17: Área de utilizador

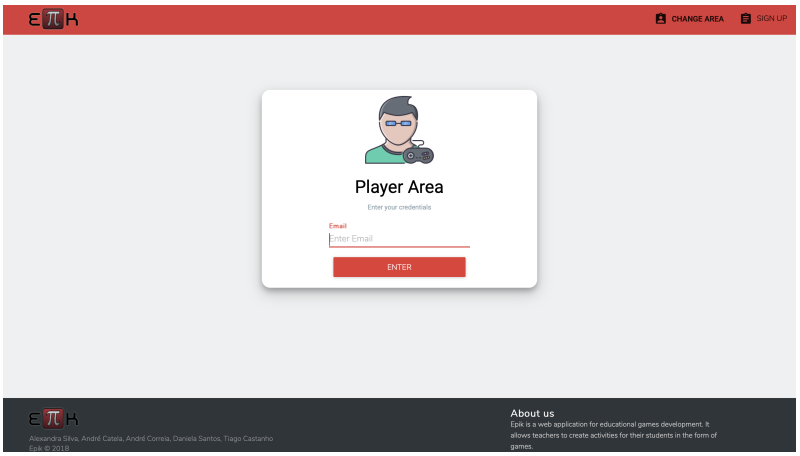


Figura 3.18: *Login* de jogador

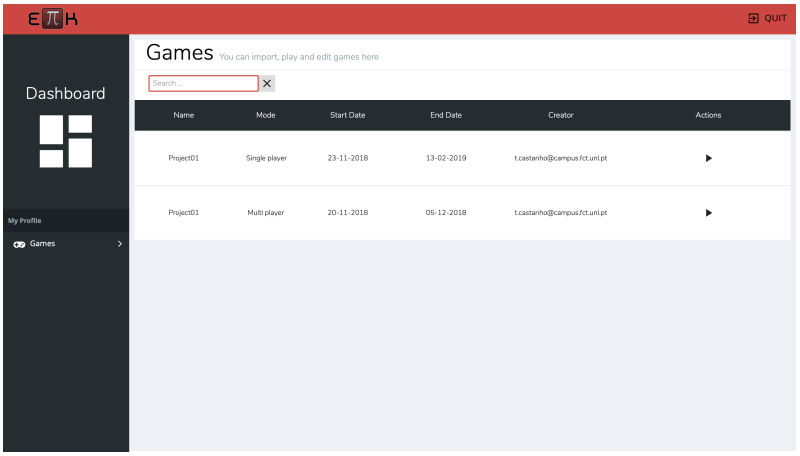


Figura 3.19: *Dashboard* do jogador

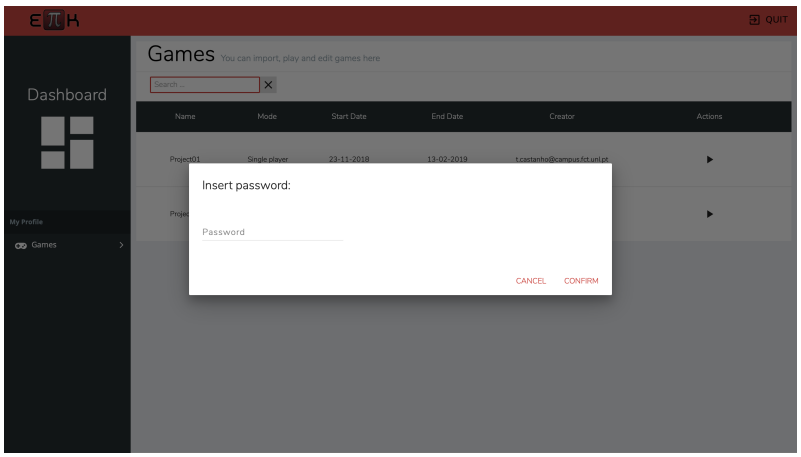


Figura 3.20: Pedido de password do jogo

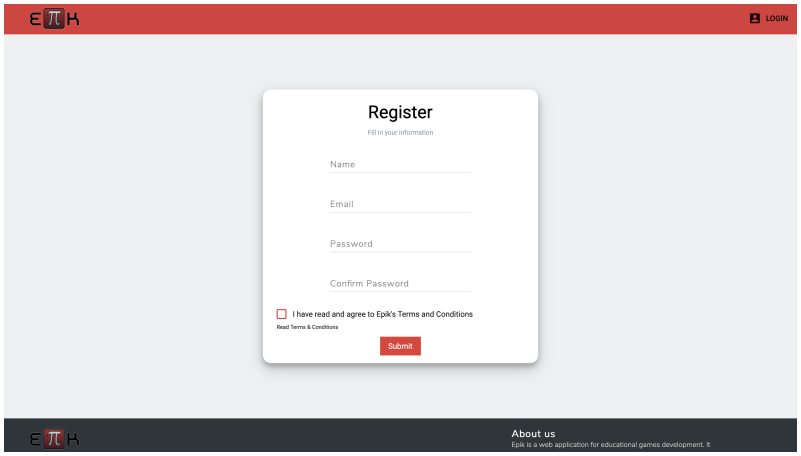


Figura 3.21: Registo de utilizador

partilhar os seus jogos com outros utilizadores e jogadores da plataforma.

Após ser utilizador registado, é possível fazer *login* selecionando a opção *developer* no menu anterior (Figura 3.17) e o *login* para esta área (Figura 3.22) requer palavra-passe, para além do email já pedido no *login* da área de jogador. Caso o utilizador se tenha esquecido da sua palavra-passe é possível pedir uma nova através do *link* no canto inferior direito do formulário e um email com uma nova palavra passe será enviado para o endereço inserido. De seguida, caso a combinação de email e palavra-passe seja verificada com sucesso pelo sistema, o utilizador entra na plataforma e a *dashboard* de programador da Figura 3.23 é apresentada ao utilizador.

A *dashboard* tem duas áreas principais: a página dos jogos e o editar perfil. Na página para editar perfil, apresentada na Figura 3.24, o utilizador pode atualizar o seu nome e mudar a sua palavra passe, no entanto para esta ultima atualização é necessário a confirmação com a palavra passe atual.

A página do servidor onde são disponibilizados os jogos é a *dashboard* do utilizador

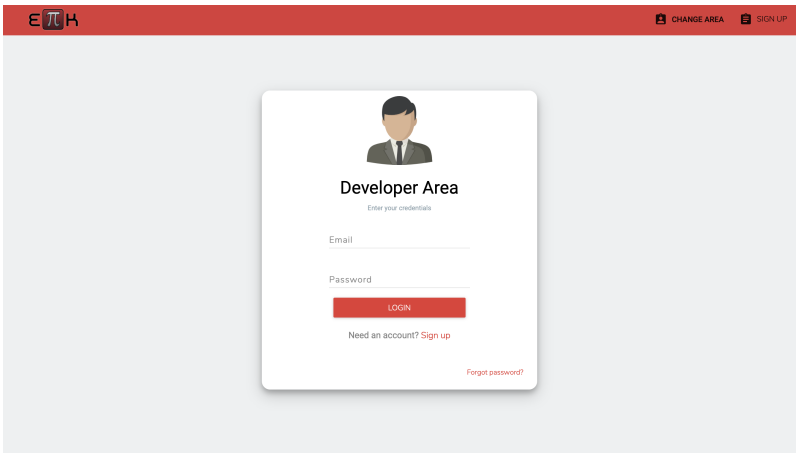


Figura 3.22: *Login* de programador

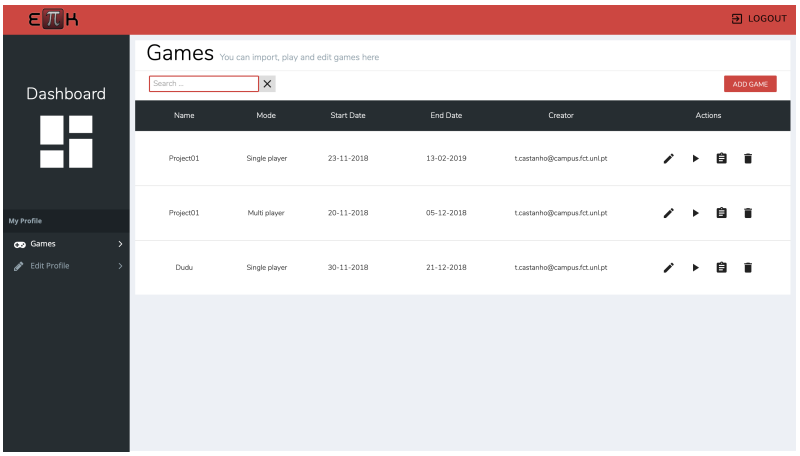


Figura 3.23: *Dashboard* de programador

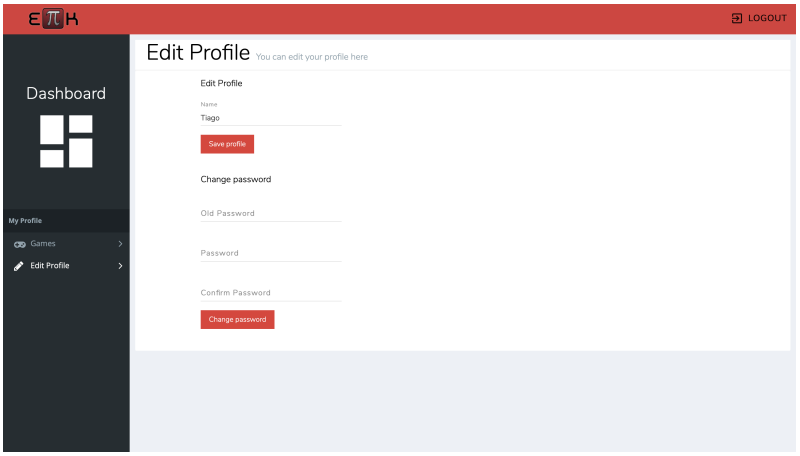


Figura 3.24: Editar perfil

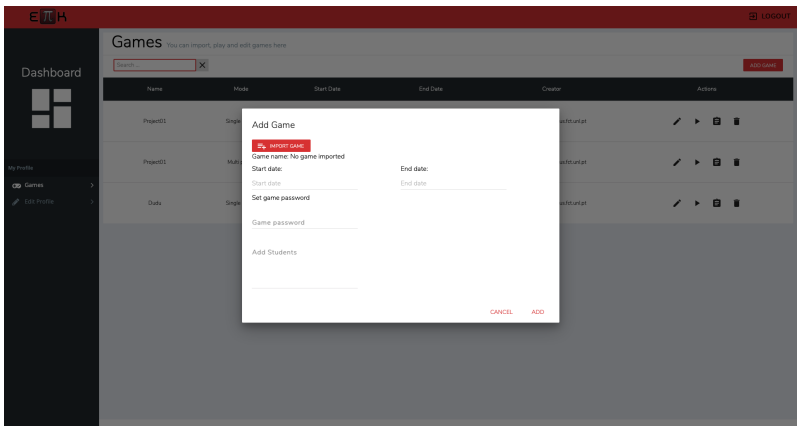


Figura 3.25: Adicionar jogo Epik

(Figura 3.23). Esta *dashboard* tem bastantes semelhanças em relação à *dashboard* do jogador referida anteriormente, esta, para além das funcionalidades descritas e de representar todos os jogos partilhados com o utilizador, a *dashboard* do utilizador permite a importação de novos jogos Epik no formato de ficheiros .ZIP.
Aquando da importação do jogo, o utilizador preenche o formulário 3.25 onde especifica as datas de inicio e termino do jogo, a palavra-passe e os emails dos jogadores com quem quer partilhar o jogo.

Após concluída a importação do jogo com sucesso, o jogo fica imediatamente disponivel na *dashboard* do utilizador e na *dashboard* daqueles com quem o utilizador partilhou o jogo, onde para além da possibilidade de jogar os jogos, o criador do jogo tem as opções de editar, ver os registos e apagar o jogo.
Ao editar o jogo, é possível alterar o nome, descrição do jogo, alterar o modo de jogo entre *singleplayer* e *multiplayer*, editar as datas de inicio e fim do jogo, criar uma nova palavra-passe e partilhar o jogo com mais jogadores. Os registos do jogo (Figura 3.26), apenas preenchidos na versão *singleplayer*, disponibilizam a lista de jogadores que já jogaram o jogo, a data em que a sessão foi realizada e ordenando a lista pelas pontuações obtidas. Para além dos registos referentes ao jogadores também é possível verificar a popularidade do jogo e saber quantas vezes este jogo foi jogado pelas pessoas com quem foi partilhado. Caso o utilizador ache que o jogo está desatualizado, não tem mais interesse ou simplesmente quer manter a sua *dashboard* minimalista pode, a qualquer momento, apagar o jogo podendo sempre voltar a importar o mesmo ou uma versão atualizada gerada posteriormente na aplicação *desktop* de desenvolvimento.

Na subsecção seguinte será descrito em maior detalhe o percurso por um jogo Epik quando o utilizador clica no ícone para jogar sabendo, à partida, que o jogo é igual para os jogadores e para o criador, apenas com a necessidade de inserção da palavra-passe definida pelo criador do jogo por parte dos jogadores.

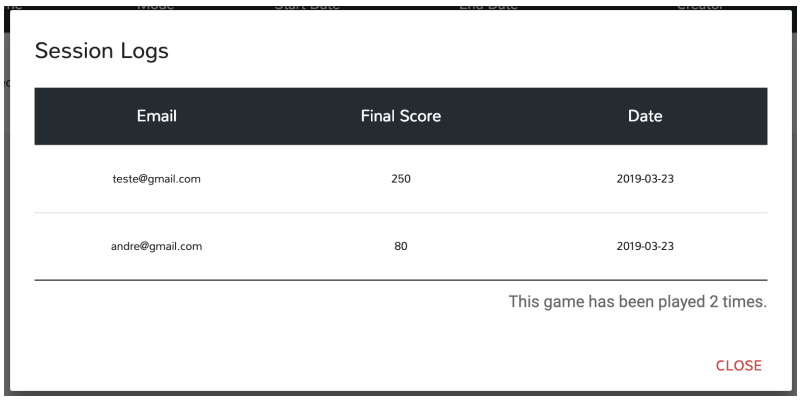


Figura 3.26: Modal com o log dos jogos

3.4.3 Execução de jogos Epik

Como referido em secções anteriores, à entrada num jogo, com ou sem palavra-passe, ou seja, criador ou jogador a estrutura do jogo é igual para ambos tendo sempre um grupo de cenários comuns a qualquer outro jogo Epik sendo estes referidos na Figura 3.1:

- **Inicio** - o cenário inicial onde é apresentado o titulo do jogo;
- **Instruções** - o cenário onde podemos consultar as regras dos jogos Epik;
- **Sala de espera** - o cenário onde o jogador define o seu avatar e espera por outros colegas em caso de jogo *multiplayer*;
- **Cenários de corpo** - o conjunto de cenários definidos pelo criador do jogo onde serão resolvidas as atividades ou consultados recursos;
- **Fim de jogo** - o cenário final caso o jogo não seja concluído com sucesso;
- **Sala de espera classificações** - cenário em que, caso o jogo seja *multiplayer*, os jogadores esperam pelo resultado final antes de aceder às classificações;
- **Classificações** - o cenário final onde é apresentada a pontuação final e, em caso de jogo *multiplayer*, os *rankings* da sessão após conclusão do jogo com sucesso.

Para além dos já existentes na versão antiga da plataforma e caso o jogo seja *multiplayer*, foi adicionado um novo cenário comum, no entanto caso o jogador seja o ultimo a acabar o jogo não o irá ver e será diretamente redirecionado do ultimo cenário de corpo para o cenário de **Fim de Jogo** ou de **Classificações** dependendo das transições definidas pelo criador do jogo. Este novo cenário funciona como sala de espera entre os cenários de corpo e os cenários finais do jogo, porque para entrar no cenário de **Classificações** é necessário saber o desempenho final de cada jogador para obter uma tabela de resultados correta.

Figura 3.27: Cenário de Início do jogo Epik

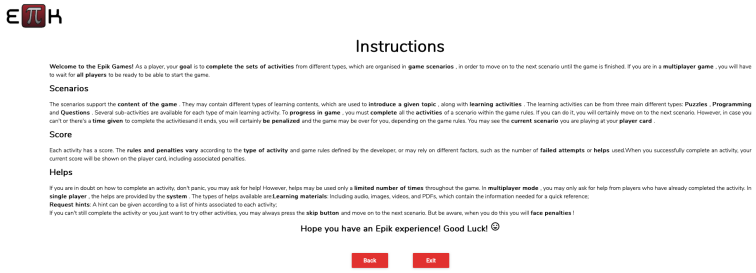


Figura 3.28: Cenário com as Instruções do jogo Epik

Após a entrada no jogo com sucesso, é mostrado o cenário de **Inicio** (Figura 3.27). Neste cenário para além do nome do jogo ser o cabeçalho da página é dada ao jogador três opções:

- ***Start***;
- ***Instructions***;
- ***Exit***;

Se o jogador selecionar a primeira opção e decidir começar o jogo, passa ao cenário seguinte do fluxo que é a **Sala de Espera** (Figura 3.29) onde poderá definir o seu nome e escolher um dos diferentes avatares para ser representado no jogo. Ao clicar na segunda opção o cenário de **Instruções** (Figura 3.28), outro dos cenários comuns, é apresentado e as instruções gerais para os jogos Epik são descritas ao utilizador. Como este é um cenário meramente informativo, a única opção existente aqui é o ***Back*** que devolve o utilizador ao cenário de **Inicio**.

A terceira e ultima opção disponível no cenário de ***Inicio*** é um botão de ***Exit*** que faz com que o utilizador volte para a sua *dashboard* de modo a escolher outro jogo para jogar

Figura 3.29: Sala de espera do jogo Epik

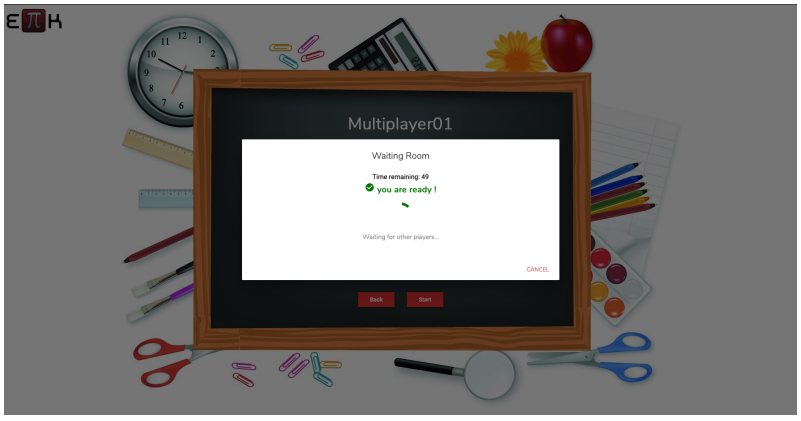


Figura 3.30: Sala de espera do jogo *Multiplayer*

ou realizar qualquer uma outra das ações descritas anteriormente na secção 3.4.2.

Prosseguindo no fluxo do jogo, o cenário seguinte é a sala de espera da Figura 3.29. Neste cenário, o jogador pode escolher o seu avatar, através dos ícones de navegação, de uma seleção de catorze disponíveis e definir o seu *nickname* pelo qual vai ser representado ao longo do jogo, no mapa do jogo e nos cartões de jogador. Ao decidir que está pronto para começar o jogo uma de duas situações acontecem, dependendo do modo do jogo. Se o jogo for do tipo *singleplayer*, o jogador é informado de que o jogo iniciará dentro de instantes e após um curto tempo de espera é redirecionado para o primeiro cenário de corpo do jogo. Caso o jogo em questão seja do tipo *multiplayer* um modal de espera (Figura 3.30) é aberto mostrando a informação dos jogadores que estão a entrar no jogo e um tempo limite para iniciar a execução do jogo. Caso o número mínimo de jogadores não seja atingido a sessão é fechada. Caso o número máximo de jogadores seja atingido antes do final do tempo limite a execução do jogo começa imediatamente com a transição para os cenários de corpo.

Iniciando os **cenários de corpo** (Figura 3.31) é possível verificar a existência de três áreas distintas na apresentação do mesmo. Estas três áreas são comuns a todos os **cenários de corpo** e apresentam informações sobre o estado do jogo e do cenário em questão. Estas três áreas são:

- Cabeçalho;
- Área de jogo;
- Cartões de jogador;

No cabeçalho é sempre apresentado ao jogador o logo do Epik, o nome do cenário onde se encontra e, por vezes, se o utilizador tenha definido para aquele cenário é também apresentado nesta área o tempo limite para execução deste cenário. Caso o tempo limite chegue ao fim o jogador é redirecionado para o cenário seguinte de acordo com a transição definida para o limite de tempo.
Na área de jogo são disponibilizados os recursos e atividades definidos pelo utilizador aquando da criação do jogo. Os recursos disponibilizados podem ser formas (quadrados, círculos, parágrafos, balões ou títulos) ou então do tipo multimédia (áudio, vídeo, imagens e PDFs), estes recursos multimédia são localmente no servidor quando o utilizador faz a importação do jogo. Caso o cenário contenha atividades, um ícone correspondente ao tipo de atividade aparecerá no local e após um clique é aberto um modal com a atividade a realizar.
Os cartões de jogador disponibilizam informação acerca do estado do jogo individualmente para cada jogador. Para além de mostrarem o nome e o avatar que escolheram, indicam a pontuação individual de cada jogador dividida em pontos de penalização, pontos de recompensa e total de pontos, indicam em que cenário se encontra o jogador que é referido nesse cartão e o estado do jogo. O jogador pode estar em três estados diferentes:

- ***Playing*** - quando o jogador está ativo no jogo está a realizar cenários de corpo;
- ***Finished*** - quando o jogador já acabou o jogo, está na sala de espera das classificações à espera dos colegas para terminar o jogo;
- ***Give Up*** - caso o jogador retorne à *dashboard* através do botão de *exit*;
- ***Disconnected*** - caso o jogador tenha desligado o computador ou encerrado o *browser*;

Por fim, existem três botões sempre presentes durante a execução dos cenários de corpo. Estes botões servem para realizar ações sobre o jogo ou para obter mais informação sobre o estado do jogo.

- ***Exit*** - para o utilizador fechar o jogo e voltar à sua *dashboard*;

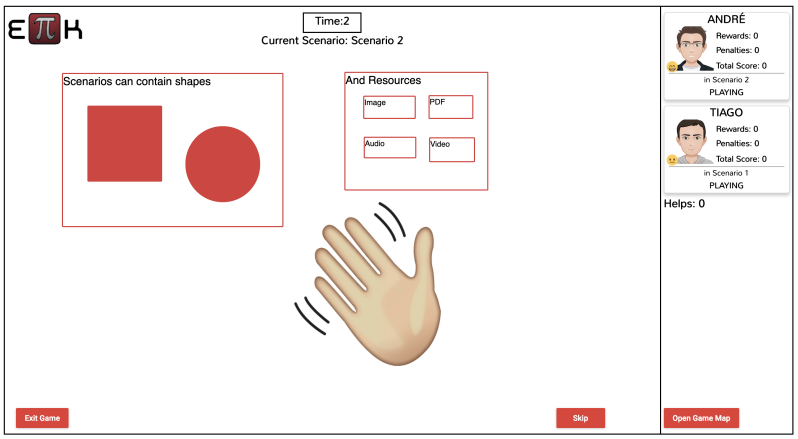


Figura 3.31: Cenário de corpo do jogo Epik

- ***Skip*** - passar o cenário à frente e realizar a transição correspondente ao *skip*. Este botão pode aparecer desativado caso o utilizador não tenha definido nenhuma transição.

- ***Open Game Map*** - Abre o mapa do jogo para obter mais *feedback* sobre o mesmo.

O mapa do jogo da Figura 3.32 apresenta ao jogador uma vista geral de todos os cenários e transições existentes entre eles, assim como os cenários já realizados e o cenário atual. Ao abrir o modal é notável certas parecenças com a interface gráfica para a construção dos grafos no ambiente de desenvolvimento com o mesmo *placeholder* e botões para gerar o grafo.
Ao gerar o grafo podemos verificar também o mesmo esquema de cores para definir as transições e o mesmo mecanismo de organização do grafo, sendo possível ao jogador utilizar o *scroll* para aumentar ou diminuir o tamanho do grafo, arrastar nós do grafo para os ordenar de uma melhor forma que o *layout* automático, passar com o rato por cima do nó para saber a descrição do cenário e passar com o rato por cima de uma aresta do grafo para saber que tipo de transição em questão. Este grafo difere do grafo existente no ambiente de execução porque não permite a criação de novas transições nem a eliminação das mesmas, indica a vermelho o cenário atual em que o jogador se encontra e indica a verde os cenários que o jogador já realizou. Caso o jogo seja *multiplayer* é fornecido uma tabela por cima do botão de *Generate Graph* que disponibiliza a informação sobre cada jogador, mostrando o cenário onde se encontra e respetiva pontuação.

Os dois cenários restantes para terminar o jogo são os cenários de **Classificações** e de **Fim de Jogo**. Os jogadores fazem a transição para estes cenários quando chegam ao final dos cenários de corpo e executam a transição que o criador do jogo definiu para estes cenários. Geralmente o cenário de **Fim de Jogo** é visto como cenário de insucesso e o cenário de **Classificações** visto como cenário de sucesso na conclusão do jogo.

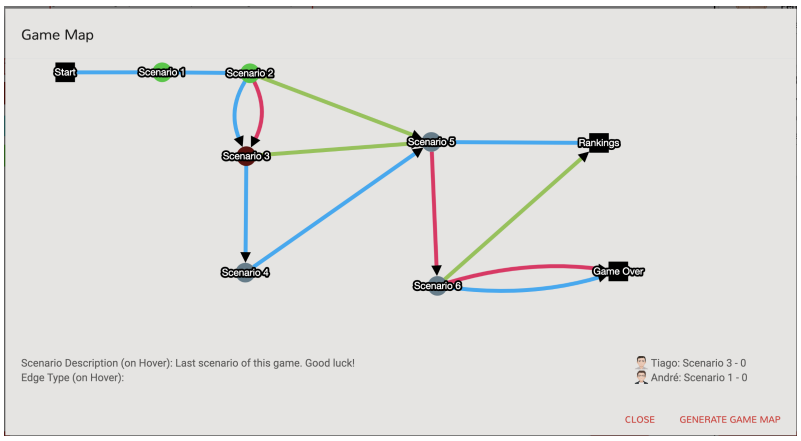


Figura 3.32: Modal com o mapa do jogo

No cenário de **Fim de Jogo** da Figura 3.33, para além de uma mensagem de final de jogo e de um ícone apresentados ao jogador, são também mostradas três opções. O jogador pode fazer ***Exit*** e retornar á sua *Dashboard*, pode fazer ***Try Again*** e retornar ao cenário de início do jogo, ou então pode querer consultar a **Classificação** final do jogo e obter mais *feedback* sobre a sua pontuação.
O cenário de **Classificações** tem duas variantes, uma para os jogos *multiplayer* (Figura 3.36), outra para os jogos *singleplayer* (Figura 3.34) e em ambas as versões são dadas ao utilizador as opções de ***Try Again*** para voltar ao cenário de ínicio do jogo ou de ***Exit*** para voltar à sua de *Dashboard* jogador. Caso o jogo em questão seja *singleplayer* é apresentado no cenário o resultado do jogo descriminando a pontuação bónus, de penalização e resultado final. No jogo *multiplayer* é necessário que já não exista nenhum jogador em jogo de modo a fornecer os dados finais na tabela final de classificação. Para isso existe uma sala de espera, como representado na Figura 3.35 após o término do jogo.
Ao terminar o jogo, todos os jogadores têm um estado final que informa o sistema de como o jogo foi terminado para aquele jogador:

- ***Finished*** - quando o jogador termina o jogo no cenário de Classificações;
- ***Game Over*** - quando o jogador termina o jogo redirecionado para o cenário de Fim de Jogo;
- ***Give Up*** - quando o jogador desiste do jogo e retorna à *dashboard* através do botão de *Exit*;
- ***Disconnected*** - quando o jogador perde a ligação ao servidor de execução;

Quando todos os jogadores terminam o jogo a base de dados é atualizada com com os resultados da sessão e é apresentada a tabela de pontuação final, é ordenada através da pontuação final de cada jogador, utilizando o tempo de execução do jogo e o estado final do jogador como critérios de desempate. Ao terminar o jogo é possível reiniciar através do botão de ***Try Again*** ou retornar à *Dashboard* através do ***Exit***.

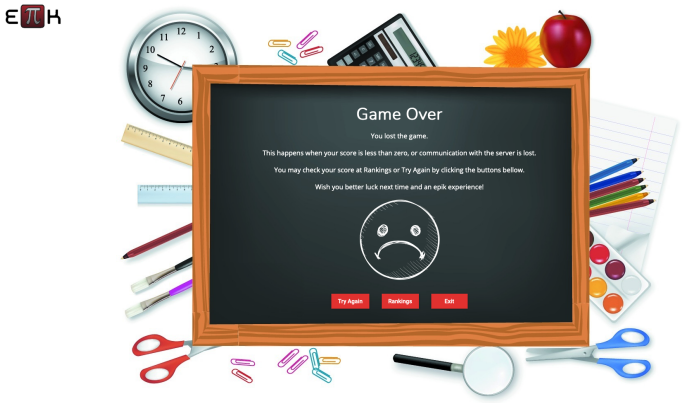


Figura 3.33: Cenário de *Game Over*

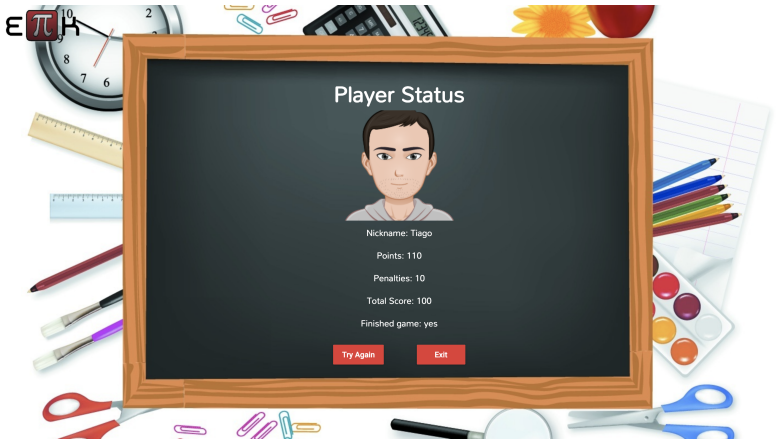


Figura 3.34: Classificações para o jogo *Singleplayer*

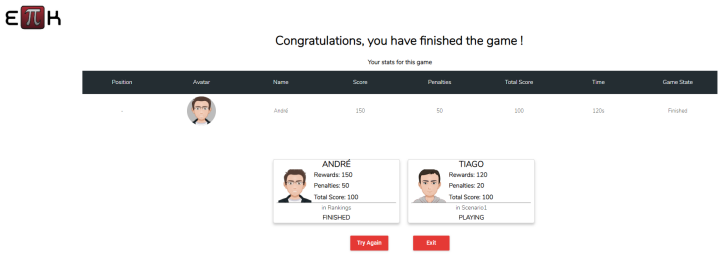


Figura 3.35: Sala de espera para final de jogo Epik

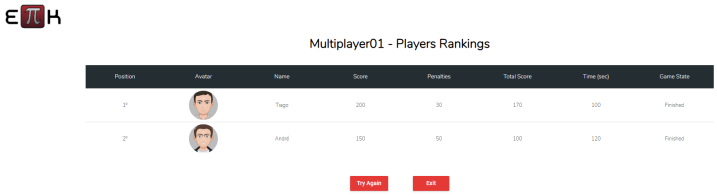


Figura 3.36: Classificações para o jogo *Multiplayer*

3.5 Sumário

Ao longo deste capítulo foram apresentadas propostas de solução para colmatar as falhas da versão anterior do Epik, efetuar a reestruturação da plataforma e implementação da componente gráfica para criação, edição e visualização de grafos.
Nos capítulos seguintes será apresentado o estudo das ferramentas de modo a efetuar uma escolha consciente e que permita obter o máximo da plataforma mantendo a coesão entre servidor de execução e aplicação *desktop*.

Capítulo 4

Análise e seleção de tecnologias

Para que fosse possível a implementação da nova plataforma Epik com funcionalidades de aplicação desktop e a funcionalidade que permite criação, edição, e visualização de grafos foi necessário realizar um estudo sobre quais as tecnologias adequadas a utilizar e integrar na plataforma. Deste modo, foi possível averiguar que tecnologias se adequariam melhor ao grupo de trabalho e às necessidades exigidas pela plataforma.

4.1 *Frameworks* de desenvolvimento de aplicações

Com o objetivo de tornar o Epik numa aplicação *desktop*, foram estudadas várias *frameworks* para que fosse possível escolher a qual se adequava melhor ao grupo de trabalho envolvido no desenvolvimento da aplicação. Apesar das tabelas (4.1 e 4.2) apresentarem uma comparação entre várias *frameworks*, nesta secção só serão referidas as três mais adequadas ao grupo de trabalho com base nos seguintes fatores:

- Fácil de usar
- Licença paga
- Linguagens suportadas

- **React Desktop** - É uma biblioteca construida por cima da biblioteca React do Facebook, pode ser usado com qualquer projeto JavaScript usando várias ferramentas do MacOS e do Windows e é baseado em componentes, o que permite a construção de interfaces mais complexas, evita redundância no código ao reutilizar os componentes se necessário e são compatíveis com sistemas operativos móveis (iOS e Android), o que permite ao programador desenvolver as suas aplicações para vários sistemas

Tabela 4.1: Tabela comparativa de *frameworks* para desenvolvimento de aplicações *desktop*

Framework	Suporte Base de Dados	Pago
React Native	Sim	Não
Electron	Sim	Não
Haxe	Sim	Não
Node Webkit	Sim	Não
Xojo	Sim, mas complicado	Sim
Enyo	Sim	Não
8th	Sim	Sim

Tabela 4.2: Tabela comparativa de linguagens e plataformas suportadas das *frameworks* para desenvolvimento de aplicações *desktop*

Framework	Linguagens suportadas	Plataformas suportadas
React Native	JS, React	Windows/Mac OS /Android/iOS
Electron	JS, HTML e CSS	Windows/Mac OS/Linux
Haxe	Android, PHP, Python, C#, Java, NodeJS	Windows/Mac OS /Android/iOS
Node Webkit	NodeJS	Windows/Mac OS/Linux
Xojo	Xojo (similar ao VB, Java e C#)	Windows/Mac OS/Linux
Enyo	JS, HTML	Windows/Mac OS /Android/iOS
8th	8th	Windows/Mac OS/Linux /Rasp/Android/iOS

diferentes sem despender tempo extra para migrar os componentes.
O React atualmente é usado por grandes empresas conhecidas mundialmente como Facebook, Skype, AirBnB e Tesla mas também por *startups* que estão a começar a desenvolver os seus projetos [18].

- **Electron** - Em 2013 era necessário uma framework para suportar o novo editor de texto do GitHub, Atom, para isso foi desenvolvido o Electron e desde então tem sido adotado tanto por *startups* como por empresas de renome. É uma framework open source, que usa Chromium e node.js para ser possível construir a aplicação com HTML e JavaScript e permite criar aplicações para vários sistemas operativos (Linux, MacOS e Windows). O Electron trata automaticamente dos *updates*, *crashes*, *debug*, menus e notificações facilitando assim o trabalho do programador.
  Algumas das aplicações mais conhecidas desenvolvidas com o Electron são: Slack, Atom, GitHub Desktop e Discord, sendo que atualmente já a App Store do MacOS (desde Maio 2016) e a Windows Store (desde Agosto 2016) suportam aplicações desenvolvidas em Electron para download [19].

- **Xojo** -Anteriormente conhecido como Real Studio, o Xojo 1 foi lançado em 2013 e continua ligado à linguagem BASIC, com base numa interface *drag-and-drop*,o Xojo, é bastante apelativo e intuitivo. No entanto para poder exportar uma aplicação é

necessário uma licença paga, definir as suas bases de dados pode ser mais complicado do que aparenta o resto da aplicação e ao contrário dos IDE normais, o Xojo, não guarda o código fonte em ficheiros de texto, mas sim num único ficheiro. Algumas das empresa que utilizam o Xojo como *framework* de desenvolvimento das suas aplicações são: Apple, Google, NASA, Intel e Cisco [20].

Tendo em conta todas as *frameworks* estudadas, os fatores referidos no inicio da secção e as funcionalidades pretendidas para a aplicação, a *framework* selecionada pelo grupo de trabalho foi o Electron pois as linguagens suportadas estavam de acordo com as competências técnicas do grupo.

4.2 Ferramentas de visualização de grafos

Após estudar e selecionar a *framework* a utilizar para o desenvolvimento da aplicação *desktop* é necessário escolher a ferramenta para criação, edição e visualização de grafos tendo em conta a compatibilidade com a *framework* escolhida na secção anterior e que permitisse ao utilizador interagir com o grafo.
De modo a representar os cenários do jogo Epik e suas transições é necessário escolher uma estrutura que permitisse ao utilizador uma fácil visualização de todos os cenários já criados e como se ligavam entre si. Esta organização do jogo em cenários pode ser vista como um grafo, onde os vértices do grafo servirão para representar cada cenário do jogo e as arestas as transições entre os mesmos. De modo a visualizar e construir o grafo foram estudadas algumas ferramentas de visualização de grafos para que fosse possível escolher a que melhor se adaptava aos requisitos propostos, sendo estes: a facilidade de integração, a linguagem em que é desenvolvido o grafo e a interatividade com o utilizador.

4.2.1 GraphViz

O GraphViz, é um software *open source* lançado em 1991 para visualização de grafos de modo a representar informação como por exemplo diagramas e redes. A ideia é transformar um simples texto (linguagem DOT) em diagramas (Figura 4.1) que possam ser mostrados noutros documentos ou páginas web e permite ainda uma vasta personalização com opções de cores, fontes, estilos, formas e links. O GraphViz não foi criado de forma a substituir o Visio e a sua utilização mais comum é em forma de *plugin* de modo a que outras aplicações consigam realizar a visualização de grafos [21].

4.2.2 Linguagem DOT

O DOT (Figura 4.2) gera os grafos orientados (Figura 4.3) a partir de hierarquias e pode ser executado através da linha de comandos, com uma interface gráfica compatível (GraphViz) ou com um serviço de visualização na web e a linguagem é uma forma de descrever grafos através de texto sendo normalmente ficheiros usando a extensão .gv ou .dot. Esta

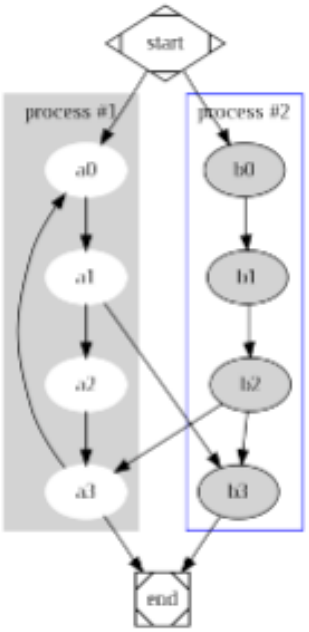


Figura 4.1: Grafo construido com GraphViz

```
digraph graphname {
    a -> b -> c;
    b -> d;
}
```

Figura 4.2: Exemplo de código para grafo orientado

linguagem tem três tipos principais de objetos: Grafos, nós e arcos, sendo possível também atribuir formatação aos objetos do grafo como por exemplo cor, forma, tamanho e estilo [22]. A construção do grafo através da linguagem DOT tem quatro fases:

- Desfazer quaisquer ciclos que existam no input.
- Atribuir cada node a um *rank* ou nível. Definir coordenada Y.
- Ordenar os nodes dentro dos *ranks* de modo a evitar cruzamentos.
- Definir as coordenadas X para manter os arcos curtos.

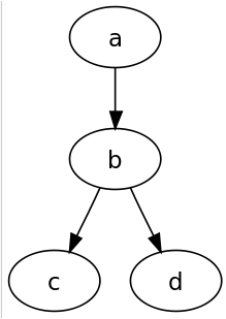


Figura 4.3: Grafo orientado

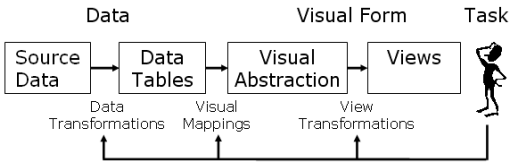


Figura 4.4: Processo de construção Prefuse

4.2.3 Prefuse

É um conjunto de ferramentas que permitem a visualização de dados de forma interativa, baseado em Java e usando a sua biblioteca gráfica 2D, está licenciado através de uma licença BSD (Berkeley Software Distribution) e pode ser usado que para efeitos comerciais e como não comerciais. O processo de construção através do Prefuse está representado na Figura 4.4 e tem como principais funcionalidades [23]:

- Componentes para tamanho, cores e formas.
- Grafos suportam atributos, *indexing* e *queries*.
- Pode conter animação.
- Linguagem parecida ao SQL e suporta *queries* para bases de dados SQL.
- Simples e APIs fáceis de usar.

4.2.4 Boost Graph Library e Graph Toolkit for Algorithms and Drawings

Sendo uma biblioteca C++ , um dos objetivos do Boost Graph Library é permitir o acesso à estrutura do grafo sem mostrar detalhes da sua implementação. Esta interface é aberta, o que permite que qualquer outra biblioteca que a implemente será compatível com os algoritmos do BGL e outros que utilizem a mesma interface, visto que todos os componentes gerados por esta biblioteca e toda a sua interface é genérica. O Graph Toolkit for Algorithms and Drawings é uma tentativa de construir implementações reusáveis de implementações de grafos e algoritmos que são implementados em classes e formalizados

em conceitos. Foi desenvolvido com os mesmos princípios que a BGL pelo que é possível usar partes do GTAD, como o grafo, em conjunto com o BGL, mas de momento , a documentação do GTAD não se encontra completa [24].

4.2.5 Cytoscape.js

Esta é uma biblioteca *open source* especifica para teoria de grafos, o Cytoscape.js, está escrito em JavaScript e pode ser usado para análise e visualização de grafos, com uma integração fácil nas aplicações *desktop browsers* assim como dispositivos móveis.
O Cytoscape [25] já tem uma API muito completa, com programação orientada a objetos, o que reduz o trabalho do programador ao incorporar esta ferramenta na sua aplicação e dando ao utilizador a possibilidade de personalizar as formas, cores e arcos desenhados e adicionar animação ao grafo.
Para melhorar a experiência do utilizador e facilitar o programador, o Cytoscape, inclui vários gestos para aplicações *desktop*:

- Arrastar fundo para alinhar grafo;
- *Pinch* no *trackpad* para fazer zoom;
- Usar roda do rato para fazer zoom;
- Usar dois dedos no *trackpad* para fazer zoom;
- Clique para selecionar;
- Clique no fundo para remover seleção;
- Múltipla seleção com a tecla modificadora do teclado (*shift*, *command*, *control*, *alt*) e clique;
- Criar uma caixa de seleção com a tecla modificadora e clique mais movimento do rato;
- Arrastar nós do grafo;

Um dos exemplos apresentados no site oficial está representado na Figura 4.5. A *performance* quebra com o aumento da complexidade do grafo, tanto em número de nós, como em personalização, no entanto são disponibilizadas dicas na documentação para colmatar estas falhas e garantir uma melhor experiência e *performance* [25].

4.2.6 Visualgo

Concetualizado em 2011 com o objetivo de ajudar os estudantes a visualizarem e entenderem melhor as estruturas de dados é um projeto em andamento e a sua melhor funcionalidade é o gerador de questões, que permite aos alunes testarem os seus conhecimentos [26].

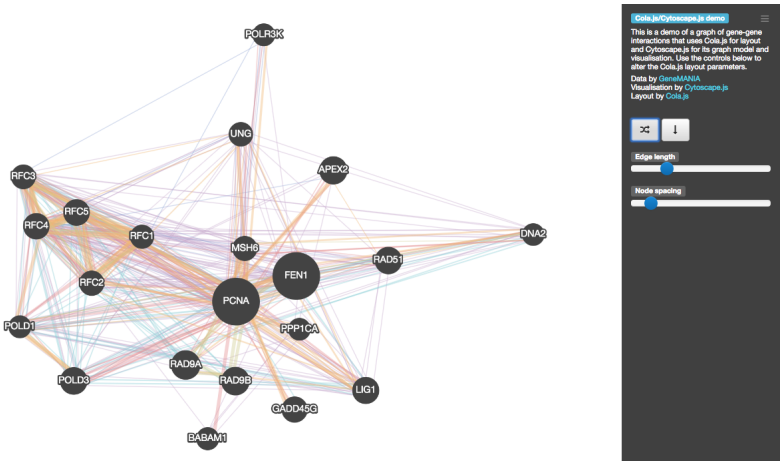


Figura 4.5: Grafo criado com Cytoscape.js

4.3 Componente gráfica de fluxos de cenários

Na aplicação de desenvolvimento de jogos, para implementar a componente gráfica de criação de fluxos de cenários será utilizado o separador de *Flow* localizado na área de trabalho (Figura 2.4) e ao criar o grafo cada cenário de corpo de um jogo Epik será correspondente a cada um dos nós. Cada um dos mecanismos de transição terá uma cor correspondente nas arestas do grafo apresentado na área de trabalho.
Para além dos quatro mecanismos já referidos no capitulo 2.3.1, será implementado um mecanismo de evolução com base na percentagem de respostas corretas que o jogador obteve no cenário, permitindo também que os jogadores presentes no mesmo jogo estejam em cenários diferentes, algo que não era possível anteriormente, tornando assim o jogo diferente de jogador para jogador e mais adaptado aos conhecimentos de cada um.
No servidor de execução de jogos será apresentado aos jogadores uma versão estática do onde será possível verificar os cenários já realizados, o que falta percorrer e também onde se encontram os outros jogadores no caso de estarmos num jogo *multiplayer*.
Então, para selecionar a ferramenta a utilizar é necessário que seja integrável no Electron, permita interatividade com o utilizador e seja personalizável sendo algumas das funcionalidades esperadas pela ferramenta: a construção do grafo com cores definidas para cada tipo de transição, criação de transições interagindo apenas com o grafo, atualização do grafo automaticamente durante a edição do projeto e visualização do fluxo aquando execução do jogo.
Para desenvolver esta componente gráfica será usado o Cytoscape.js [25] (referido na secção 4.2.5) porque permite uma fácil incorporação na plataforma, tem gestos já definidos para aplicações *desktop*, é interativo e existem várias opções de personalização.

4.4 Sumário

Neste capitulo é apresentado o estudo das ferramentas de modo a escolher as mais adequadas a utilizar na implementação do projeto. Podemos concluir que existem algumas ferramentas gratuitas às quais os elementos do grupo são mais familiares no caso das *frameworks* de desenvolvimento de aplicações, no entanto para a visualização de grafos a maior parte das ferramentas não permitem a interatividade necessária para as funcionalidades desejadas.

Capítulo 5

Implementação

A nova plataforma Epik é composta por uma aplicação desktop e um servidor de execução de jogos. Para a implementação destas duas componentes foram utilizadas tecnologias semelhantes de modo a manter a coesão ao longo de todo o desenvolvimento do projeto.
Para ambas as componentes foi utilizado React, composto por JavaScript, HTML5 e CSS3, para a implementação das interfaces e lógica, assim como o Cytoscape.js para construção, edição e visualização de grafos na componente individual desta dissertação. Na camada de dados é onde surge uma maior diferença nas tecnologias utilizadas, pois, na aplicação de desenvolvimento foi utilizado o SQLite [27] que permite a gestão da base de dados sem ligações externas através de um ficheiro guardado na raiz do projeto, enquanto que no servidor de execução foi utilizado o MySQL [14] permitindo vários acessos em simultâneo através da rede.
A aplicação desktop guarda toda a informação sobre os projetos e atividades criadas naquela instalação da aplicação e o servidor de execução permite a importação de jogos e sua execução por vários jogadores em simultâneo.

5.1 Arquitetura da aplicação *desktop*

Durante a implementação, a arquitetura da aplicação foi dividida em três camadas como apresentado na Figura 5.1 a camada de apresentação, a camada lógica e a camada de dados.
Durante a utilização da plataforma, o utilizador interage diretamente com a camada de apresentação na *dashboard*, criação de projetos, criação de atividades e importação de recursos. Esta camada foi desenvolvida utilizando React composto por JavaScript, CSS3 e HTML5 respeitando as regras e os temas do Material-UI da Google [28].

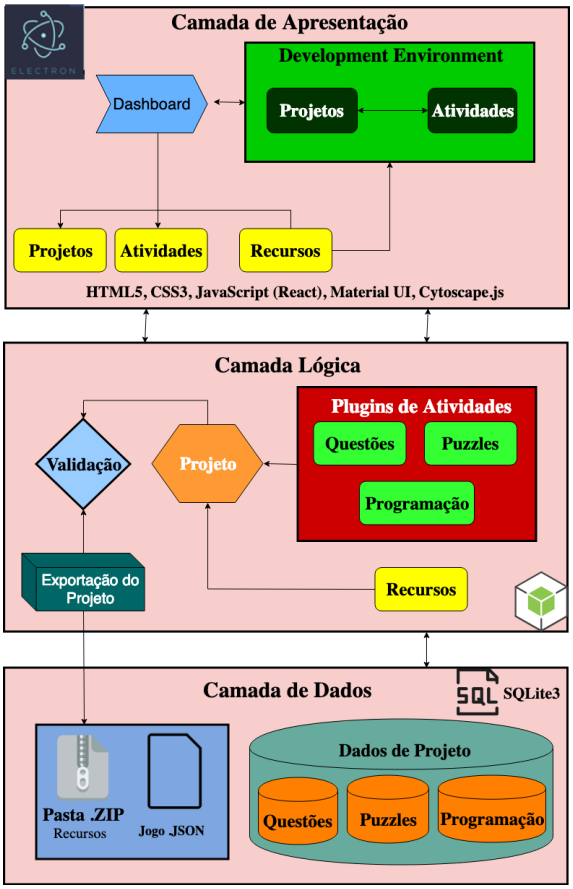


Figura 5.1: Modelo de arquitetura da aplicação

A camada lógica é constantemente atualizada consoante as ações realizadas na camada de apresentação, permitindo a ligação com a camada de dados. Na camada lógica é feita a validação dos projetos e atividades no momento de exportação e a ligação entre atividades e recursos com os projetos aos quais estão designados.

Para a implementação da camada de dados da aplicação foi utilizado o SQLite que permite guardar a base de dados num ficheiro localmente, estando sempre disponível *offline* mantendo a aplicação sem necessidade de conexão à internet. Nesta camada é, também, possível acedar aos ficheiros .Zip gerados para cada projeto no momento de exportação.

Esta divisão em três camadas é implementada e incorporada na *framework* escolhida para desenvolvimento de aplicações *desktop* Electron, que permite uma fácil integração da base de dados em SQLite3, React, Interact.js e outras *frameworks* necessárias para o desenvolvimento das componentes individuais de cada elemento do grupo de trabalho, como o *Cytoscape.js*. Por fim, e para que fosse possível exportar a aplicação para um executável foi utilizado o Electron-builder que permite a exportação para MacOS, Windows e Linux, no entanto a versão para Linux não se encontra completa devido à falta de algumas funcionalidades apenas compatíveis em MacOS e Windows

5.1.1 Camada de apresentação

A camada de apresentação da aplicação está organizada em quatro componentes nos quais o utilizador interage diretamente: a *dashboard*, recursos, projetos e atividades, existindo dois ambientes de desenvolvimento próprios para os projetos e atividades.
Após o estudo das *frameworks* existentes foi escolhido o Electron devido às tecnologias suportadas. Então, para a implementação das quatro componentes foi utilizado React, baseado em JavaScript, HTML5 e CSS3 que para além de serem tecnologias recentes, todos os elementos do grupo de trabalho estavam familiarizados à partida com a sua utilização e permite uma coesão com o servidor de execução visto serem tecnologias muito utilizadas no desenvolvimento web. Outra das vantagens pelas quais o Electron foi escolhido como *framework* a utilizar foi o facto de permitir fácil integração com ferramentas externas como o Interact.js, Cytoscape.js e SQLite.
O Interact.js é utilizado na área de desenho da aplicação no momento de construção de cenários. Esta ferramenta permite-nos a utilização de *drag-and-drop* e *resize* em div's implementadas em HTML5, assim como o desencadear de funções quando o utilizador realiza alguma ação sobre um elemento do cenário.
O Cytoscape.js está implementado no separador de *flow* presente na área de desenho permitindo nesta secção a criação, edição e visualização de grafos, foi escolhido pois das ferramentas estudadas na secção 4.2 era a que permitia uma maior interação com o utilizador, uma vasta personalização e a sua linguagem de implementação é igual à utilizada no Electron o que permite uma fácil integração na *framework*.

5.1.2 Camada lógica

Composta por quatro componentes, a camada lógica processa as ações do utilizador na plataforma e as componentes correspondentes.
É composta pelos recursos e atividades, que podem ser puzzles, questões ou de programação, que por sua vez formam os projetos. Durante o ato de guardar um projeto, a camada lógica efetua validações para determinar se o projeto é válido e está pronto a ser exportado na forma de .ZIP, de modo a evitar inconsistências no momento de importação do jogo para o servidor de execução. Estas validações são descritas com mais detalhe na secção 5.4.1

5.1.3 Camada de dados

A camada de dados é composta por uma base de dados implementada em SQLite3 e os .ZIP gerados no momento de exportação do jogo, em que durante a utilização da plataforma a base de dados SQLite3 é constantemente requisitada pela camada lógica de modo a serem apresentados aos utilizadores valores consistentes.
O SQLite permite gerir uma base de dados sem ligação à internet, apenas através de um ficheiro na raiz do projeto, e são armazenados todos os dados relacionados com uma

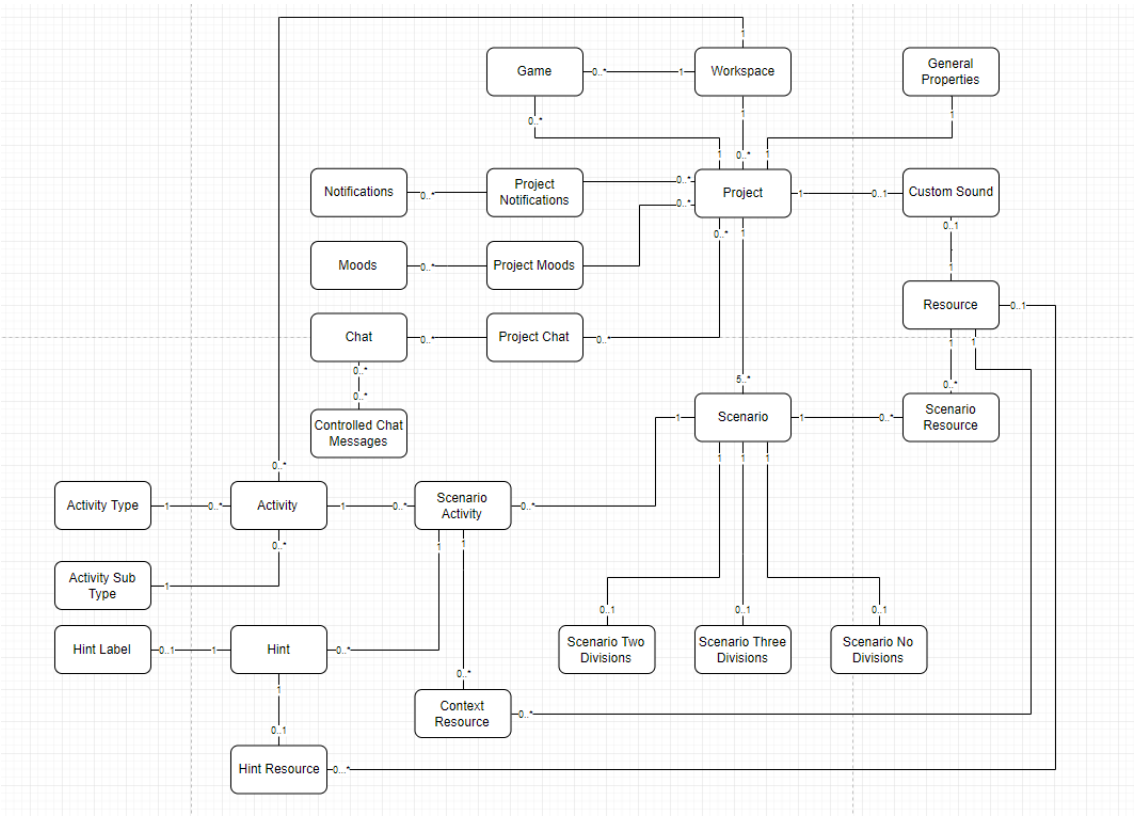


Figura 5.2: Modelo de dados para ambiente de desenvolvimento de jogos Epik

instalação da aplicação. Durante o processo de exportação de um jogo Epik e após verificar que o projeto é válido, a camada lógica replica todos os dados de um projeto e guarda num ficheiro JSON, permitindo assim uma leitura mais eficaz por parte do servidor de execução, e de seguida armazena este JSON num ficheiro .ZIP com todos os recursos utilizados no projeto em questão.

5.1.4 Modelo de dados do ambiente de desenvolvimento

O modelo de dados da base de dados utilizada é apresentado na Figura 5.2. A implementação foi feita utilizando SQLite que permite a criação de bases de dados locais, sem a existência de conexões, e individuais em que cada um dos computadores em que a aplicação *desktop* para desenvolvimento de jogos está instalada.
Na base do modelo de dados está o *Workspace* tendo a respetiva tabela duas relações para com os Projetos e Atividades, permitindo assim uma melhor organização dentro da aplicação através da utilização de diferentes *Workspaces* de forma, se desejar, a organizar o seu conteúdo por temas.
De seguida e como centro do modelo de dados está a tabela dos Projetos. A partir desta tabela o modelo de dados é guardada toda a informação necessária para a geração do jogo Epik e para isso são estabelecidas relações com as seguintes tabelas:

- ***Workspace*** - Indica qual dos *workspaces* criados pelo utilizador é que está a ser

usado para armazenar o projeto, servindo de forma de organização caso o utilizador decida utilizar esta funcionalidade;

- ***General Properties*** - Estas propriedades, sempre relacionadas com o projeto, são guardadas de modo a manter coesão ao longo de todo o projeto e execução do jogo. São guardadas informações a nível estético dos cenários que estarão presentes do inicio ao fim do jogo como, a posição do logo ou as cores, bordas e textos da barra de jogadores, e propriedades para, caso o jogo seja do tipo colaborativo, saber quando iniciar o jogo com a informação guardada em *MinPlayers* e *MaxPlayers*.

- ***Custom Sound*** - Com a possibilidade de edição dos som de resposta correta e sendo este comum ao longo do projeto é necessário ter a informação acerca desse som como o caminho do ficheiro, nome, valor (que nos permite ativar ou desativar o som) e o identificador de recurso que nos permite voltar a reutilizar esse som num outro projeto ou cenário.

- ***Scenario*** - A tabela de cenários guarda sempre informações de pelo menos cinco cenários para cada projeto, sendo estes os cenários comuns à execução do jogo. Em cada cenário é possível definir a sua cor de fundo, nome e descrição mas para a correta execução do jogo e transição entre cenários os quatro principais atributos da tabela são:

  - ***UseKnowledge*** - Define se vai ser usado o conhecimento por base das transições. Caso utilize, as transições utilizadas serão definidas pelo atributo seguinte, caso contrário utilizará as transições da tabela *Scenario No Divisions*;
  - ***KnowledgeDivisions*** - Após o utilizador decidir se deseja transições com base no conhecimento, pode ainda decidir se quer duas ou três divisões e caso o valor deste atributo seja 2 a tabela utilizada será a *Scenario Two Divisions*, caso seja 3 utilizará a *Scenario Three Divisions*;
  - ***isStart*** - Define qual o cenário inicial do jogo após a execução passar pela sala de espera. Durante a validação do jogo é garantido que apenas existe um cenário com o atributo ***isStart*** a 1 por cada projeto;
  - ***Time*** - Este atributo guarda o limite de tempo que o jogador tem no cenário antes que o tempo seja dado por terminado e execute uma transição do tipo *timeover*.

  Para além das tabelas de transição referenciadas acima, a tabela dos cenários tem também relações com as tabelas das atividades e dos recursos que por sua vez guardam o estilo, conteúdo e tipos dos recursos ou atividades em questão.

- ***Game*** - No momento de exportação do jogo guarda informação do projeto de modo ao utilizador obter um sumário do jogo que já gerou tendo fácil acesso a informações como o modo de jogo, a data da sua exportação, nome e descrição.

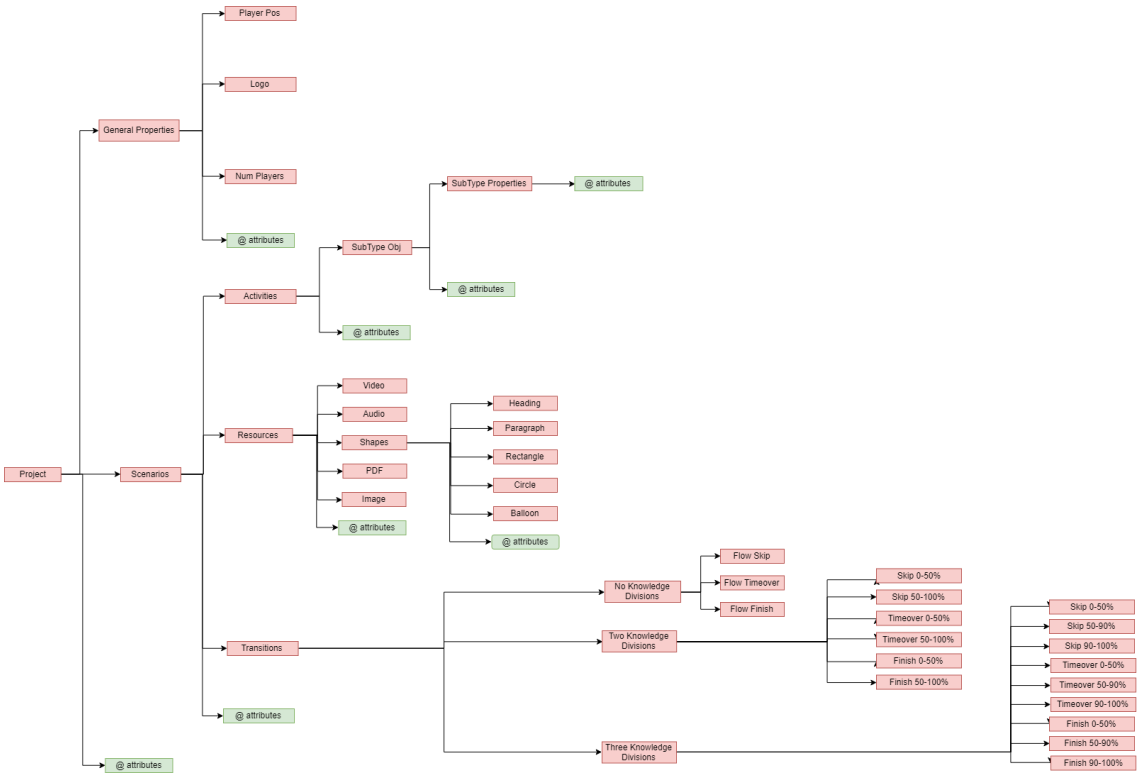


Figura 5.3: Formato do ficheiro de jogo Epik

- ***Project Chat***, ***Project Mood*** e ***Project Notifications*** - Estas tabelas guardam a informação sobre como os jogadores podem interagir entre si através dos *Moods* e do *Chat*, assim como as notificações que o jogo pode enviar a cada um dos jogadores.

5.1.5 Formato do ficheiro de jogo Epik

Os ficheiros de jogos Epik servem para transitar informação sobre o jogo do ambiente de desenvolvimento para o servidor de execução, a sua estrutura é representada na Figura 5.3 e o formato utilizado para o ficheiro é o .JSON. Neste ficheiro é possível consultar as propriedades gerais do projeto, os vários cenários constituintes e o conteúdo destes cenários, quer sejam recursos, atividades ou as transições para este definidas.

Na raiz do ficheiro encontra-se sempre o nó do projeto sendo toda a informação guardada no ficheiro em relação ao projeto em questão. Nos níveis seguintes do ficheiro do jogo estão guardados os seguintes dados:

- ***Attributes*** - descrição geral do jogo com por exemplo: o nome, tipo de jogo, data de criação e descrição mais detalhada escrita pelo criador do jogo sobre o seu conteúdo;
- ***General Properties*** - guarda a informação geral do projeto como o número de jogadores, as posições do painel de jogadores e do logo do Epik, e alguns atributos

visuais sobre cores e texto;

- ***Scenarios*** - neste nível do ficheiro existem os atributos gerais de cada cenário como o nome, a descrição, a data de criação, a cor de fundo, se requer ou não conhecimento para a execução de transições e o tempo definido pelo utilizador para a realização do cenário. Para além de destes atributos gerais existem ainda três atributos cujo o seu conteúdo está referido em níveis seguintes da estrutura do ficheiro, sendo estes:

  - ***Activities*** - onde são armazenadas as propriedades inseridas no cenário referido, indicando os atributos sobre o tamanho, pontuações e posição no cenário, o tipo de atividade como puzzle, questões ou programação e caso seja necessário indica também o subtipo da atividade. O conteúdo desta informação varia dependendo do tipo da atividade em questão, visto que cada atividade necessita de dados diferentes para ser executada;

  - ***Resources*** - os recursos constituintes do cenário tem os dados sobre a sua posição e tamanho no cenário, nome e tipo. Caso o tipo seja um dos recursos multimédia existe o atributo sobre o caminho do ficheiro de modo a carregar o seu conteúdo com sucesso, caso contrário e o recurso seja uma forma geométrica, os atributos incluem informações extra sobre o estilo do recurso como cores, bordas, sombreados, tamanho e alinhamento do texto que pode ou não existir;

  - ***Transitions*** - o conteúdo guardado neste nível do ficheiro depende dos atributos *useknowledge* e *knowledgedivisions* definidos nos atributos dos cenários. Caso o atributo de *useknowledge* seja 0, então as transições só guardaram informação para três tipos diferentes: *Skip*, *Timeover* e *Finish*.
    Caso contrário, o número de informações guardadas depende do atributo *knowledgedivisions* onde o utilizador definiu no ambiente de desenvolvimento se deseja que existam duas ou três divisões com base no conhecimento. Assim sendo, caso o utilizador tenha definido que deseja o duas divisões com base no conhecimento existirão seis atributos, onde os três referidos anteriormente serão divididos em frações de 50% ficando assim com: *Skip* entre 0% e 50%, *Skip* entre 50% e 100%, *Timeover* entre 0% e 50%, *Timeover* entre 50% e 100%, *Finish* entre 0% e 50% e *Finish* entre 50% e 100%.
    Então, se o utilizador definir três divisões por conhecimento para o cenário, os três atributos passarão a ser nove divididos de 0% a 50%, de 50% a 90% e de 90% a 100%, premiando assim deste modo a excelência do aluno e guardando informação para as seguintes transições: *Skip* entre 0% e 50%, *Skip* entre 50% e 90%, *Skip* entre 90% e 100%, *Timeover* entre 0% e 50%, *Timeover* entre 50% e 90%, *Timeover* entre 90% e 100%, *Finish* entre 0% e 50%, *Finish* entre 50% e 90% e *Finish* entre 90% e 100%.

Em qualquer um destes atributos a informação guardada é sempre o identificador do cenário de destino a partir do cenário em questão no nível superior do ficheiro.

Após a importação com sucesso do ficheiro .ZIP que contém o ficheiro do jogo Epik é possível jogar o jogo através de leituras diretas ao ficheiro em questão devido a esta estrutura e ao formato escolhido. Assim, com esta decisão foi possível simplificar a base de dados do servidor de execução não guardando alguns dos dados presentes no ficheiro e melhorar o desempenho do servidor evitando leituras desnecessárias à base de dados no momento de execução do jogo.

5.2 Arquitetura do servidor de execução

O servidor de execução, localizado na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, e sua respetiva arquitetura (Figura 5.4) é composta por quatro camadas, em semelhança com a arquitetura da aplicação *desktop* descrita na secção 5.1. É composta também por uma camada de apresentação, camada lógica e por uma camada de dados, todavia foi adicionada uma camada de comunicação entre a camada de apresentação e a camada lógica de modo efetuar pedidos através da internet. No entanto, o utilizador, para ter acesso ao servidor de execução é necessário conectar-se à rede da Faculdade ou ter as credenciais válidas para acesso por VPN à rede interna.
Na camada de apresentação é disponibilizado o *website* Epik e a área de jogo onde os utilizadores poderão jogar e importar jogos caso sejam utilizadores registados.
Na camada de comunicação o utilizador interage com o servidor através de pedidos feitos pelo Socket.IO [15] e recebe as respostas às suas ações pelo mesmo meio.
Na camada lógica os jogos importados são geridos e distribuídos pelos jogadores com quem o utilizador partilhar o jogo e durante a execução do jogo organiza os dados da sessão do jogo gerando os registos para o criador do jogo.
Na camada de dados foi utilizado MySQL para a construção da base de dados pois era uma tecnologia à qual todos os elementos do grupo estavam familiarizados e os jogos são importados no formato de .ZIP compostos pelo JSON que tem os dados do jogo, os recursos e atividades utilizados nos diversos cenários que compõem o jogo Epik em questão.

5.2.1 Camada de apresentação

Dividida por dois componentes, a camada de apresentação é composta pelo *Website* Epik e pela área de jogo em que ambos os componentes foram desenvolvidos com recurso ao HTML5, CSS3, e JavaScript, utilizando também componentes do Material-UI. Esta camada interage diretamente com a camada de comunicação através de pedidos e recebendo as atualizações de modo a apresentar aos utilizadores informação coerente e consistente.
No *website* é possível fazer o *download* da aplicação *desktop*, dos *plugins* de atividades,

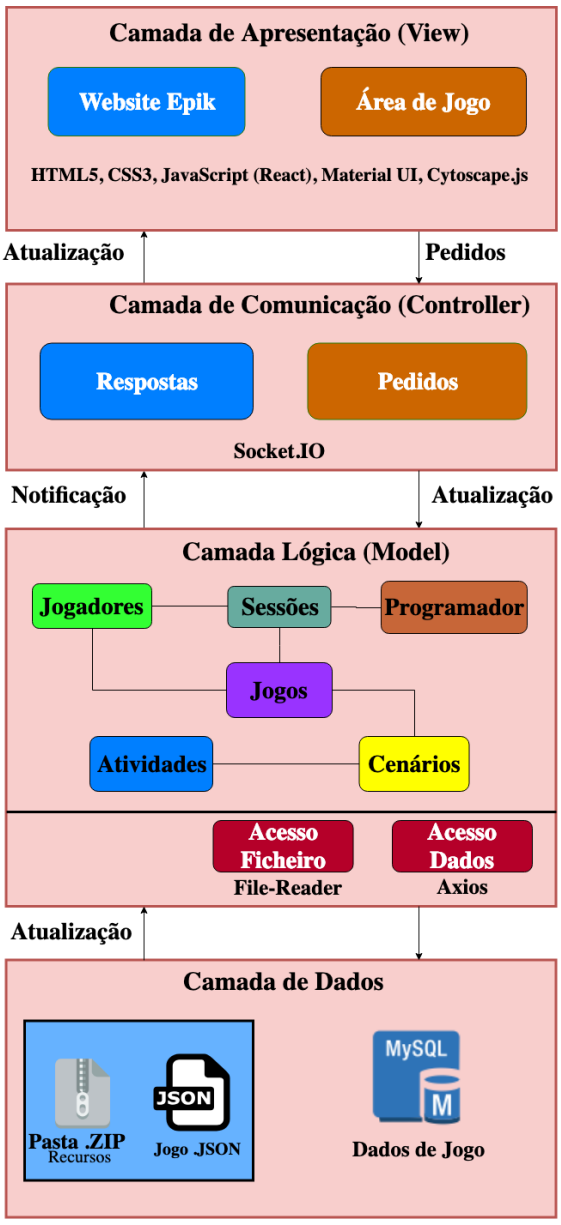


Figura 5.4: Modelo de arquitetura do servidor de execução

obter informações sobre a plataforma e entrar na área de jogador onde é possível criar uma conta para poder importar e partilhar jogo, fazer *login* na sua conta ou entrar como jogador onde não é necessário registo e aceder a todos os jogos que foram partilhados com o seu email.
Após entrar na sua *dashboard*, os utilizadores registados podem editar os seus dados, importar e partilhar jogos ou jogar um dos jogos disponíveis.

5.2.2 Camada de comunicação

Durante a utilização do servidor de execução, a comunicação é realizada através de pedidos identificados por um URL, definidos na classe de *routes* e compostos por objetos em formato JSON caso o pedido em questão seja uma inserção na base de dados. Estes pedidos são realizados através da biblioteca *Axios* [29] que facilita a definição de pedidos a bases de dados MySQL, sendo apenas necessário definir os dados de inicio de sessão para com o MySQL num ficheiro de configuração na pasta de origem do projeto e os URL que identificam cada pedido.
O Socket.IO [15] referenciado na Figura 5.4 é utilizado na criação de sessões de jogo *multiplayer* e através desta biblioteca é possível a ligação externa de vários jogadores a uma só sessão. Com a utilização desta biblioteca foi-nos possível implementar ambas as salas de espera presentes no formato *multiplayer*, percebendo quando existe o número máximo e o número mínimo de jogadores conectados a uma sessão de jogo, e permitir que os jogadores ligados através da mesma sessão não percam a ligação mesmo estando em cenários diferentes do jogo, fazendo com que sigam o fluxo de cenários independentemente do resto dos jogadores em sessão.

5.2.3 Camada lógica

A camada lógica do servidor de execução é composta por cinco principais componentes que comunicam entre si como representado na Figura 5.4. No centro da camada lógica estão os jogos que interligam todos os outros componentes através de si. Os jogos são compostos por cenários que por sua vez são compostos por atividades tendo uma ligação direta entre si.
Um jogador fica imediatamente ligado a um jogo no momento de importação do mesmo e o seu email é indicado, no entanto apenas no momento de execução é criada uma sessão estabelecendo a ligação entre jogo e jogador que permite guardar os dados sobre a atual execução do jogo, como por exemplo as pontuações e os registos aos quais o programador tem acesso através das sessões.
Os cinco componentes necessitam de acesso à camada de dados descrita na secção 5.2.4 e de forma a comunicar com a base de dados é utilizada a biblioteca Axios. Para aceder ao ficheiro JSON guardado no servidor foi utilizado um leitor de ficheiros, representado na imagem como *File-Reader* [30]. Um exemplo em que camada lógica necessita de efetuar uma leitura ao ficheiro é no momento de iniciação de um novo jogo Epik. Neste momento

o ficheiro JSON que corresponde ao jogo selecionado é carregado através do *File-Reader* para um estado do tipo objeto que guardará esta informação até ao término do jogo. Devido a esta decisão na implementação e à utilização do Socket.IO foi possível que, nos jogos *multiplayer*, os jogadores se mantivessem conectados à mesma ligação e URL não perdendo contacto com os restantes jogadores na sessão. Sendo apenas necessário atualizar os estados necessários para a visualização do cenário através das leituras ao ficheiro referidas anteriormente no momento da transição.

5.2.4 Camada de dados

Na camada de dados do servidor de execução existem duas componentes principais: os .ZIP importados e a base de dados desenvolvida em MySQL. Os .ZIPs são gerados no ambiente de desenvolvimento e de forma a criar um jogo Epik no servidor de execução, no momento de importação, é necessário descomprimir o ficheiro e guardar os recursos e ficheiro JSON para que sejam acedidos aquando da execução do jogo.
A conexão com a base de dados é feita através da biblioteca Axios e é preenchida com dados dos utilizadores no momento do registo na aplicação, dados do jogo no momento da importação do jogo e resultados finais sobre as sessões de jogo no momento de finalização do jogo.
O ficheiro JSON e os recursos guardados no servidor são utilizados aquando da execução do jogo. O JSON é carregado para um estado no inicio do jogo e os estados necessários para a execução do jogo são atualizados através de leituras ao ficheiro com ajuda do *File-Reader*. Os recursos são acedidos quando requisitados para compor o cenário onde o jogador se encontra.
Na subsecção seguinte o modelo de dados é apresentado com mais detalhe.

5.2.5 Modelo de dados do ambiente de execução

Representado na Figura 5.5, o modelo de dados para o ambiente de execução foi implementado em MySQL e tem várias semelhanças ao modelo de dados apresentado na Figura 5.2 correspondente ao modelo de dados do ambiente de desenvolvimento, pois é necessário guardar a informação do jogo, das atividades e dos cenários. Para além desta informação que é gerada e guardada no momento de importação do jogo é necessário armazenar informações sobre os utilizadores da plataforma gerindo os seus acessos, jogadores e seu desempenho durante a execução do jogo.

- ***Users*** - Registo de todos os utilizadores registados na plataforma, apenas utilizadores inseridos nesta tabela têm a permissão para importar e partilhar jogo;

- ***Games*** - Os jogos importados pelos utilizadores guardam a sua informação básica na base de dados para uma partilha mais fácil para com os jogadores;

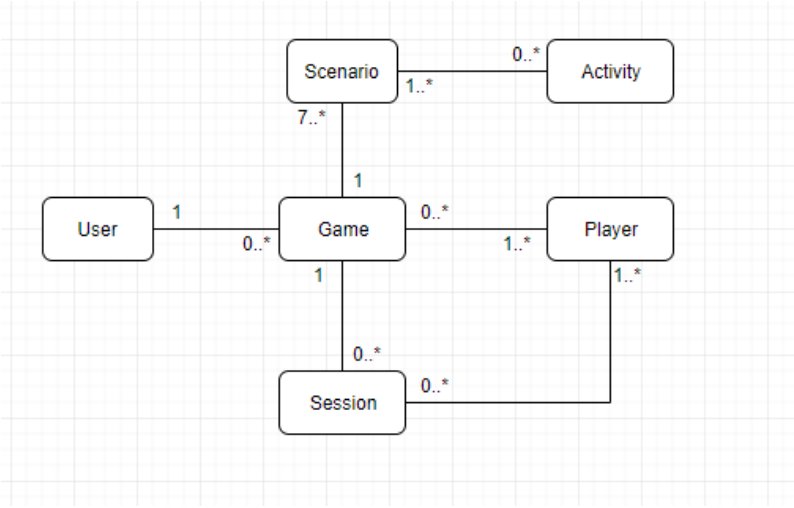


Figura 5.5: Modelo de dados do ambiente de execução

- ***Player*** - Mantêm o registo de todos os jogadores que já fizeram *login* na plataforma ou que já tem jogos partilhados consigo e estão ligados à tabela dos jogos através dos seus ids;

- ***Session*** - De modo a manter um registo dos desempenhos de todos de jogadores que jogaram cada um dos jogos importados para o servidor de execução no momento de finalização do jogo é criada uma entrada nesta tabela com o registo de uma nova sessão.

As tabelas de *scenario* e *activity* são utilizadas ambas para a criação dos registos e identificar a que jogo são pertencentes guardando assim um registo dos jogos já importados na plataforma mesmo após o ficheiro .JSON ser eliminado.

5.3 Transições de cenários

Durante a execução, após terminar um cenário, o jogador vai transitando entre cenários de modo a evoluir no jogo com o objetivo de terminar o mesmo. Então, no momento de execução de um jogo, quando o utilizador dá o cenário por terminado seja por *Skip*, *Time-out* ou por terminar as atividades, a camada lógica realiza várias verificações de modo a garantir a evolução correta no jogo por parte do jogador.
Inicialmente o método verifica se, para o estado React do tipo objeto que guarda a informação do cenário em questão, existe conhecimento na base das transições. Caso não exista, é acedido o estado, também do tipo objeto, que guarda as transições e é utilizado o identificador que está definido no tipo de transição efetuada. Caso existam transições definidas por conhecimento, é verificado ainda, no estado que guarda o objeto cenário, o número de divisões definidas pelo utilizador e consoante esse número existem diferentes grupos de transição: se existir a divisão em dois, é necessário calcular se percentagem da

pontuação obtida é maior ou menor que 50% da pontuação total do cenário, se as divisões definidas forem três é necessário calcular se a pontuação obtida é menor que 50%, se é entre 50% e 90% ou se está acima dos 90% da pontuação total do cenário. Em ambos os casos esta verificação serve para escolher o atributo correto a utilizar do estado que guarda as transições. Para efetuar o cálculo da percentagem para a qual se deve realizar a transição é verificado se o valor guardado na variável que guarda a pontuação do cenário é maior ou menor do que a multiplicação do numero de atividades presentes no cenário e a percentagem em questão, como na Listagem 5.1.

Listagem 5.1: Cálculo de percentagem no cenário

```
1 if(score>= this.state.activities.length*90)
```

Após as verificações e a escolha correta do identificador a utilizar a pontuação do jogo é atualizada, o cenário terminado é adicionado ao vetor de cenários realizados que é utilizado para atualizar o mapa do jogo e, através do objeto que guarda a informação do jogo carregada previamente através do *File-Reader* e do identificador de cenário escolhido, são carregadas as informações, para o estado do tipo objeto, para o cenário seguinte como os recursos, atividades, formas, cores, tempo limite e as respetivas transições das quais o novo cenário é origem.
Caso o jogo em questão seja *multiplayer*, o processo acima descrito é efetuado para cada jogador, o que vai permitir que exista uma evolução no jogo de forma independente sem a necessidade que todos os jogadores se encontrem no mesmo cenário, sendo que a ferramenta que permite esta implementação é o Socket.IO mantendo a ligação dos jogadores a uma sessão criada no inicio do jogo e gerindo a coexistência de estados privados e partilhados. Deste modo, as pontuações, os estados e os cenários atuais são partilhados para todos os jogadores mas cada jogador tem acesso aos seus estados privados carregando apenas a sua informação necessária para a execução de cada cenário.

5.4 Componente gráfica para construção de grafos

De modo a atingir os objetivos propostos para esta componente gráfica para construção de grafos e sua correta implementação foi necessário pesquisar e selecionar uma ferramenta que permitisse interatividade com o grafo já gerado, existisse uma documentação clara e fosse compatível com as linguagens já utilizadas ao longo do projeto. Assim sendo, das ferramentas estudadas na secção 4.2, o Cytoscape.js, foi a qual cumpriu os três requisitos referidos na frase anterior e desta forma a utilizada na implementação desta componente. De modo a obter a construção do grafo e as várias funcionalidades propostas implementadas foram usados vários métodos e eventos disponibilizados pela ferramenta em questão na sua documentação.

- ***Add*** - adiciona um elemento ao grafo. O parâmetro utilizado na função é um objeto com os atributos grupo (podendo este ser um nó ou uma aresta), dados identificadores e posição;
- ***CSS*** - substitui o estilo pré-definido para o elemento do grafo em questão. O parâmetro utilizado é um objeto com os atributos de CSS a substituir;
- ***Layout*** - define através de um algoritmo escolhido as posições para os elementos do grafo, existindo uma lista de *layouts* disponíveis e cada um com os seus atributos próprios;
- ***On*** - define e efetua funções sobre o grafo quando o evento definido ocorre num certo tipo de elemento. O método é constituído por três parâmetros em que o primeiro define o tipo de evento, o segundo o tipo de elemento onde ocorrerá o evento e o terceiro a função despoletada pela ocorrência desse mesmo evento no elemento. Os três tipos de evento utilizados na implementação das funcionalidades da componente gráfica são:
  - ***Mouseover*** - este evento irá acionar a função quando o ponteiro do rato estiver por cima do elemento indicado;
  - ***Mouseout*** - este evento irá acionar a função quando o ponteiro do rato deixar de estar por cima do elemento indicado;
  - ***Tap*** - este evento irá acionar a função quando existir um clique do rato no elemento indicado;
- ***Destroy*** - quando invocado, este método elimina o grafo de modo a poupar a *performance* da restante aplicação, evitando assim com que o Cytoscape.js esteja em execução em segundo plano.

Na implementação do grafo foi necessário definir inicialmente o estilo para os nós, diferenciando os que estão selecionados dos que não estão com um acentuar de cor, e para as arestas do grafo criando assim curvaturas de modo a impedir que arestas com o mesmo nó de origem e de destino se sobrepusessem. De seguida, através do método *add* em que são necessários o grupo do elemento que estamos a adicionar, identificador e nome, são adicionados os nós correspondentes a todos os cenários que não são comuns a todos os projetos. Estes nós são adicionados com o estilo definido anteriormente percorrendo o vetor de cenários que a função recebe como parâmetro. Depois são adicionados os três cenários comuns necessários à criação do grafo (Inicio, Fim de Jogo e Classificações) com alterações fixas no estilo de modo a facilitar a experiência e visualização por parte do utilizador, desta forma estes três cenários comuns em vez de serem nós circulares cinzentos, são nós quadrangulares pretos destacando assim dos cenários criados pelo utilizador. Para completar a construção do grafo é necessário adicionar as arestas e para isso é feita um pedido à base de dados sobre cada cenário representado no grafo. Em cada um destes

pedidos são devolvidos o número de divisões por conhecimento que estão definidas naquele cenário e os correspondentes identificadores de destino em cada uma das situações. Consoante o número de divisões implementadas naquele cenário apenas é utilizado um grupo de transições, não sendo possível utilizar transições que não requerem conhecimento com transições que requerem pelo menos 90% de respostas corretas. Assim, se existir um identificador de destino definido para a situação e com recurso ao método *add*, em que para além do identificador e do nome da aresta é necessário também definir o identificador de origem e o identificador de destino, uma aresta é adicionada com o estilo definido anteriormente alterando apenas a cor para cada situação:

- Verde - sobre transições em que o jogador termina as atividades;
- Azul - sobre transições onde o jogador faz *skip* do cenário;
- Vermelho - sobre transições onde o jogador deixa o tempo definido para o cenário acabar.

Por fim, para organizar a disposição do grafo é aplicado um *layout* de uma lista de exemplos apresentados na página web do Cytoscape.js. O *layout* escolhido foi o *klay* porque era o que melhor se adequava ao espaço disponível para a visualização do grafo permitindo ainda definir as arestas com direção e o espaçamento entre nós no momento de geração.
No entanto, com esta implementação apenas seria possível construir o grafo e orientá-lo da maneira que o utilizador quisesse, então, para implementar as restantes funcionalidades descritas foi utilizada a função *on* da ferramenta que necessita de três parâmetros: o evento que vai acionar a função, o elemento e a função a realizar. Nos eventos de *mouseover* tanto de nós como de arestas, são apresentadas na legenda do grafo as descrições do elemento que está por baixo do cursor. Estas descrições deixam de ser apresentadas quando o evento realizado sobre o elemento é o *mouseout*.
Para criação e eliminação de arestas o evento utilizado é o *tap*. Ao acionar este evento num nó do grafo, este nó fica selecionado como nó de origem e ao voltar a clicar num nó, diferente que o nó de origem, um modal é aberto apresentando os títulos dos nós selecionados e uma lista das possíveis transições entre aqueles dois nós. Ao confirmar a desejada transição, uma nova aresta é adicionada ao grafo atualizado. Se o nó de origem da nova transição seja o nó de *Start*, a informação do cenário de destino é atualizada no campo *isStart* tornando o cenário como primeiro da ordem dos cenários de corpo. Caso o elemento do evento *tap* seja uma aresta é apresentado um modal que pergunta se o utilizador deseja realmente apagar a transição em questão e caso seja confirmada a intenção, o grafo é atualizado e a aresta é eliminada do grafo.
Ao sair da componente de gráfica para a construção de grafos, para melhorar o desempenho da plataforma e libertar a área do grafo é necessário ainda a utilização da função *destroy* para eliminar o grafo.

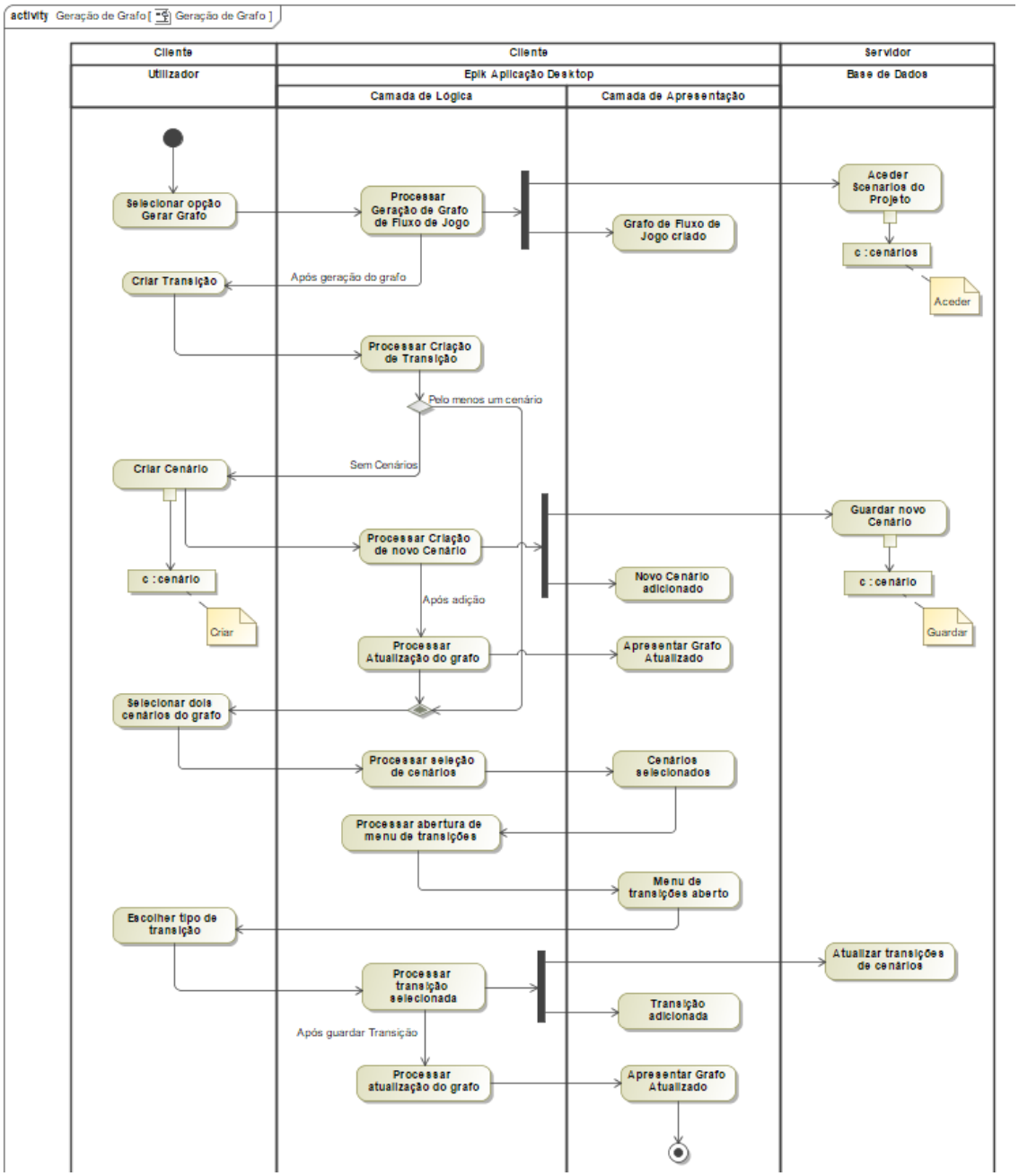


Figura 5.6: Diagrama de atividades para criação da componente gráfica de construção de grafos

Na Figura 5.6 pode ser consultado o diagrama de atividades sobre o processo de criação de uma transição entre dois cenários de um projeto em que as três camadas da arquitetura comunicam entre si para retornar o resultado final do processo, um grafo atualizado com uma nova transição escolhida pelo utilizador.

5.4.1 Validação do jogo Epik

No momento de gravação do jogo Epik,a plataforma informa o utilizador se o projeto que está a ser guardado é válido para exportação ou não. Para isso é necessário recorrer à base de dados e a algumas funções do Cytoscape.js, disponíveis na sua documentação [25], que nos vão permitir concluir se o fluxo de execução do jogo obedece às regras referidas na secção 3.2.2. Por omissão, o jogo é inicialmente valido e apenas no caso de chumbar em alguma das verificações é que é considerado inválido aparecendo a mensagem de erro correspondente no ecrã. Devido às seguintes funções do Cytoscape.js, não foi necessário a implementação de novas estruturas de dados para suportar a validação do jogo para além de um estado Booleano que guarda o resultado final da validação. Assim sendo, as funções utilizadas durante a validação:

- ***Outgoers*** - Devolve as arestas que tem como origem o nó selecionado no argumento da função e os nós destino destas arestas;
- ***Incomers*** - Devolve as arestas que tem como destino o nó selecionado no argumento e os nós de origem destas arestas;
- ***Size*** - Indica o numero de elementos presentes no objeto Cytoscape em questão. Neste âmbito é usado para verificar o numero de arestas que entram ou saem de cada nó;
- ***Predecessors*** - Aplica recursivamente a função ***Incomers*** e obtém as arestas e os nós dos quais existe um caminho definido para o nó selecionado;
- ***Contains*** - Verifica se o nó/aresta existem no objeto Cytoscape devolvendo um resultado booleano.

A primeira das verificações a ser realizada é no caso dos cenários em que tem o conhecimento por base das transições quando temos uma das divisões com transição definida, todas as restantes transições dessa divisão têm que estar preenchidas, por exemplo: se o existe uma transição definida para o *skip* e pontuação entre os 0% e os 50% então a transição dos 50% aos 100% não poderá estar definida como *NULL*. Esta verificação está representada na Listagem 5.2.

Listagem 5.2: Preenchimento por divisões

```
1 if((result[0].flowskip0_50 != null && result[0].flowskip50_100 == null)
2      || (result[0].flowskip0_50 == null && result[0].flowskip50_100 != null))
```

É necessário ao jogo ter pelo menos dois cenários criados pelo utilizador e para confirmar isso é verificado se o vetor de cenários passado para a criação do grafo tem um tamanho menor que dois.
As funções *outgoers* e *incomers*, descritas na documentação do Cytoscape.js, juntamente

com a função de *size* executam três verificações: que existe apenas um cenário de inicio, nenhum cenário volta para a sala de espera e que o cenário de Classificações tem pelo menos um nó direcionado para ele. Existem ainda mais duas funções do Cytoscape.js utilizadas, *predecessors* e *contains*, e para as seguintas duas verificações é necessário percorrer o vetor de nós e confirmar que todos os cenários têm o nó de *Start* como antecessor e que o cenário de Classificações tem todos os outros cenários na sua lista de antecessores verificando assim que é possível chegar a todos os cenários a partir do *Start* e que todos os cenários têm um caminho até ao cenário de Classificações. Estas verificações estão exemplificadas na Listagem 5.3.

Listagem 5.3: Teste grafo de fluxos

```
if (rows.length < 7) {
 this.setState({
 validGame: 0, validMessage: "The game must have at least two scenarios."
  })
     }

if (cy.$('#Start').outgoers().size() != 2
  || cy.$('#Start').incomers().size() != 0) {
 this.setState({
 validGame: 0, validMessage:"The game must have a start scenario."
   })
    }

if (cy.$('#' + rows[3].id).incomers().size() < 2) {
 this.setState({
 validGame: 0, validMessage:"At least one scenario must have a flow to the end."
  })
   }

for (var i = 6; i < rows.length; i++) {
if (!cy.$('#' + rows[i].id).predecessors().contains(cy.$('#' + rows[5].id))) {
  this.setState({
  validGame: 0, validMessage:"Every scenario must be reachable from the start."
   })
    }
   }

for (var i = 5; i < rows.length; i++) {
if (!cy.$('#' + rows[3].id).predecessors().contains(cy.$('#' + rows[i].id))) {
  this.setState({
  validGame: 0 , validMessage:"Every scenario must have a path to the end."
    })
   }
  }
```

A ultima verificação com base no grafo é o teste à sua acíclicidade, da Listagem 5.4,

adaptando o algoritmo apresentado nos diapositivos disponibilizados na unidade curricular de Análise e Desenho de Algoritmos [31] e as funções acima referidas de *incomers* e *outgoers*. Este algoritmo de teste à acíclicidade foi utilizado devido à sua simplicidade e compatibilidade com os métodos descritos na documentação do Cytoscape.js, podendo assim garantir que o jogo nunca entrará em ciclos causando a repetição de cenários. O algoritmo de teste à acíclicidade utilizado é apresentado na listagem **??**.

O algoritmo recebe como parâmetros o grafo *"cy"* e o vetor de cenários *"rows"* e cria inicialmente três variáveis auxiliares, o número de nós já processados *"numProcNodes"*, um vetor com os nós prontos a ser processados *"ready"* e um vetor que guarda o grau de entrada de cada nó *"inDegree"*. Como o cenário de inicio já foi verificado anteriormente na listagem acima apresentada podemos começar por declará-lo como pronto adicionando ao vetor *"ready"*.

O vetor de cenários é percorrido ignorando os cenários de Inicio, porque já está no vetor *"ready"*, sala de espera e instruções porque não constam no grafo, e adiciona todos os nós que tenham grau de entrada zero ao vetor *"ready"* marcando assim os mesmo como prontos a serem processados.

Enquanto o vetor *"ready"* não estiver vazio os nós serão processados. É retirado um nó do vetor com a função *pop*, o numero de nós processados é incrementado e são guardados numa variável auxiliar *"aux"* os nós para onde o nó processado está direcionado. O grau de entrada desses nós é decrementado e é verificado se atingiu o valor zero, se sim é adicionado ao vetor *"ready"* para ser processado.

Para terminar o algoritmo e determinar se existem ciclos no grafo, o numero de nós processados é comparado com o tamanho do vetor de cenários (menos os dois cenários de instruções e sala de espera) e caso estes sejam iguais podemos afirmar que o grafo gerado é acíclico.

Listagem 5.4: Teste à Aciclicidade

```
1
2
3  isAcyclic(cy, rows) {
4   var numProcNodes = 0;
5   var ready = [];
6   var inDegree = [];
7   ready.push("Start");
8    for (var v = 0; v < rows.length; v++) {
9      if (rows[v].name != "Start" &&
10      rows[v].name != "Waiting Room" &&
11      rows[v].name != "Instructions") {
12       inDegree[rows[v].id] = (cy.$('#' + rows[v].id).incomers("node").size());
13          if (inDegree[rows[v].id] == 0) {
14            ready.push(rows[v].id);
15             }
16         }
17     }
```

```
18      while (ready.length != 0) {
19          var node = ready.pop();
20          numProcNodes++;
21          var aux = cy.$('#' + node).outgoers("node");
22          for (var k = 0; k < aux.length; k++) {
23              inDegree[aux[k]._private.data.id]--;
24              if (inDegree[aux[k]._private.data.id] == 0) {
25                  ready.push(aux[k]._private.data.id);
26              }
27          }
28      }
29      return numProcNodes == rows.length - 2;
30  }
```

Por fim, já sem recurso ao grafo, é verificado que todos os cenários têm pelo menos um recurso associado, isto é, não existem cenários vazios. Esta verificação é realizada com acesso ao explorador de ficheiros e seus respetivos vetores, em que, se existir algum vetor de recursos vazio a verificação falha atualizando na base de dados confirmando que o jogo não está pronto a ser exportado.
O diagrama de atividades da Figura 5.7 representa o processo de exportação de um jogo Epik em que o processo de validação descrito acima no momento de guardar o projeto e é representado como um dos requisitos para a exportação do jogo ser concluída com sucesso, caso contrário é enviada uma notificação ao utilizador de que verificação falhou de modo a ser mais fácil a sua resolução.

5.5 Novo *feedback* para jogadores

Como descrito na secção 2.1.2, uma das principais componentes para o fluxo de execução em jogos educacionais ser bem implementado é a existência de *feedback*. Ou seja. os jogadores tem que saber o estado do jogo, o que correu bem e o que correu mal durante a sua execução.
Durante a execução das atividades esse *feedback* é fornecido através da pontuação final e é instantânea à sua realização, mas durante a realização de cenários sem atividades não existe informação para além do cenário em que o jogador e os seus colegas se encontram e suas respetivas pontuações. Então, para permitir uma visão geral sobre o jogo e o seu estado foi implementado, no painel de jogadores um botão que abrirá uma janela onde poderá ser consultado o mapa do jogo, as pontuações de cada jogador, o seu desempenho ao longo do jogo e o que falta para ser jogado.
O mapa de jogo está representado na Figura 5.8 onde é possível verificar parecenças com a componente gráfica referida na secção anterior. Este mapa foi implementado usando as mesmas funções de ***Add***, ***CSS*** e ***Layout*** descritas acima e o mesmo algoritmo percorrendo o estado que contém o vetor de todos os cenários presentes no jogo e suas respetivas transições. No entanto, comparando com a componente gráfica para construção de grafos

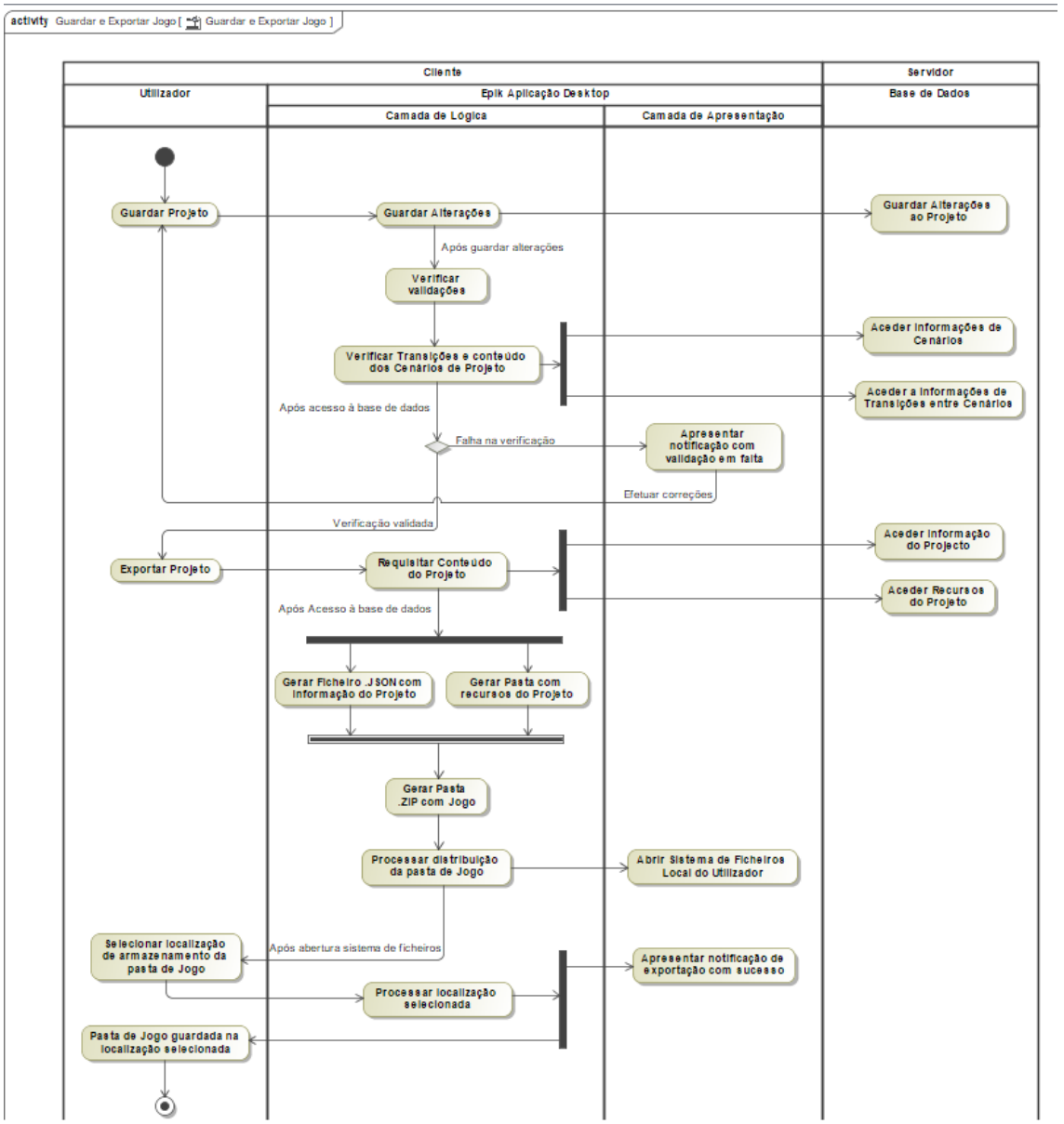


Figura 5.7: Diagrama de atividades para exportação de um jogo Epik

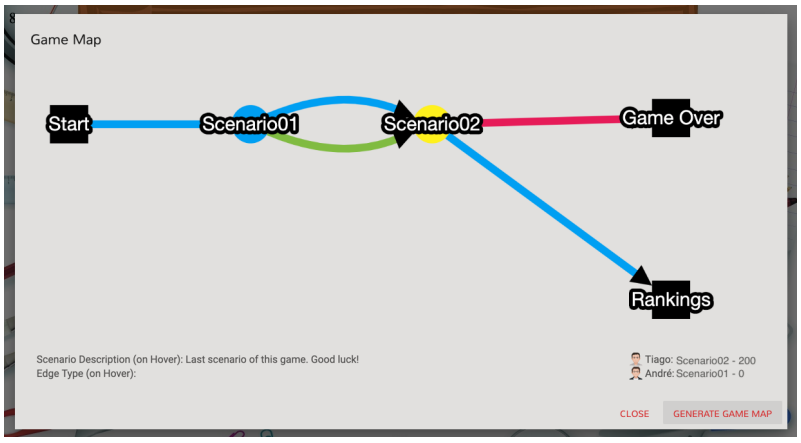


Figura 5.8: Mapa do jogo

foi removida a possibilidade de adicionar e remover arestas do grafo e adicionando um novo esquema de cores aos nós do grafo.
Na Figura 5.8, o jogador encontra-se no *Scenario02* que está representado pela cor amarela e realizou um transição do tipo *Skip* a partir do *Scenario01* e por isso este está marcado pela cor azul. Caso a transição utilizada seja de *Timeover* o cenário de origem dessa transição será representado com a cor vermelha e caso seja uma transição do tipo *Finish* será utilizada a cor verde para a sua representação. Para a implementação desta funcionalidade, no momento de adição de nós ao grafo é verificado se o cenário está numa das três listas de cenários realizados (uma lista para cada tipo de finalização de cenário) ou se o nome é correspondente ao cenário atual, e caso o nó a adicionar esteja numa destas listas ou seja o cenário atual é lhe atribuída uma cor para ser representado no grafo.
Por fim, e para complementar o mapa do jogo existe ainda uma tabela de informações que indicando a descrição do cenário aquando passagem do rato sobre um nó do grafo ou o tipo de transição caso seja uma aresta, sendo que para a implementação desta tabela foram utilizadas as funções de ***On***, ***Mouseover*** e ***Mouseout***. Caso o jogo seja *multiplayer* foi implementada uma versão simplificada do painel de jogadores informando a pontuação dos jogadores em jogo e o cenário onde se encontram.

5.6 Sumário

Neste capitulo foram descritas as arquiteturas da aplicação *desktop* e do servidor de execução de jogos Epik, assim como algumas decisões a nível de ferramentas a utilizar e implementações das partes referentes aos objetivos desta dissertação.
Ao longo desta implementação as decisões assumidas foram com o objetivo de manter a coesão entre as duas componentes do projeto, utilizando as mesmas ferramentas excetuando as bases de dados em que foi necessária a utilização do SQLite3 para a aplicação *desktop* com o intuito da sua vertente *offline*.
No capitulo seguinte são apresentados os resultados dos questionários feitos aos utilizadores da plataforma.

Capítulo 6

Avaliação

Ao longo deste capitulo são apresentados os resultados do questionário realizado aos utilizadores que testaram a aplicação desktop do Epik.
Os questionários abrangiam a parte comum e as partes individuais envolvidas no projeto, pelo que ao longo deste capítulo serão focadas as questões relacionadas com o desempenho geral da plataforma e com a utilização da componente gráfica para desenvolvimento de fluxos.

6.1 Descrição dos inquéritos e utilizadores

O objetivo da realização destes questionários, disponíveis no anexo I, foi avaliar a interface, o desenvolvimento de jogos educativos através da aplicação *desktop* e obter opiniões sobre o uso de jogos no ensino. Como um dos temas principais deste inquérito é a utilização dos jogos no ensino, os utilizadores inquiridos foram professores e explicadores de diversas áreas de e abrangendo também os vários níveis de ensino. Os inquéritos respondidos estavam compostos em oito secções, começando pela identificação do utilizador e a sua opinião sobre a utilização de jogos na educação, passando pela avaliação da interface geral da aplicação e das implementações individuais de cada membro do grupo de trabalho e terminando com uma secção sobre a sua satisfação geral para com a plataforma. As questões pertencentes ao questionário são de escolha múltipla, em que alguns casos mais que uma opção pode ser selecionada, sim ou não, resposta aberta ou de classificação de um a cinco, onde o valor cinco representa um bom resultado e o valor um representando um representando um resultado menos bom e a melhorar.

O grupo de utilizadores foi composto por vinte e um professores que após testarem a plataforma responderam ao questionário e com os resultados obtidos das primeiras quatro

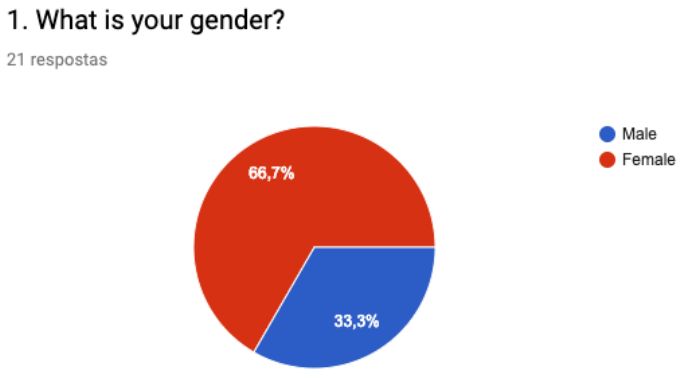


Figura 6.1: Distribuição dos utilizadores por género

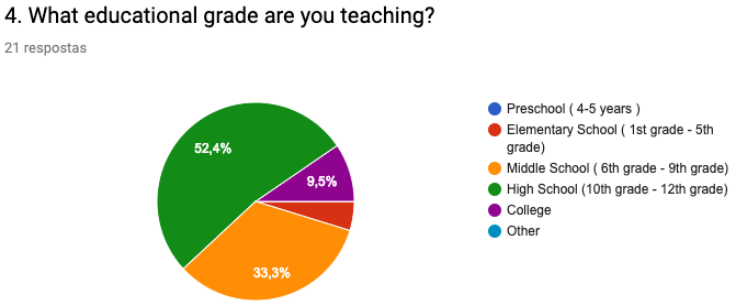


Figura 6.2: Distribuição dos utilizadores por nível de ensino

perguntas é possível identificar melhor as características do grupo de professores. Dos utilizadores catorze eram do sexo feminino e sete do sexo masculino, como apresentado no gráfico da Figura 6.1, com idades compreendidas entre os 20 e os 60 anos de idade em que cinco tinham entre os 20 e os 30 anos de idade, três entre os 31 e os 40, seis entre os 41 e os 50, e sete entre os 51 e 60. Para além do sexo e da idade para identificar os utilizadores foi questionado o nível em que lecionam, do ensino básico ao ensino superior, e há quantos anos o fazem, de um a mais de vinte anos e como seria de esperar, os resultados sobre o tempo em que lecionam estão correlacionados aos resultados da idade. O gráfico da Figura 6.2 apresenta os resultados relação ao nível que os utilizadores lecionam onde conseguimos juntar resultados de professores desde o ensino primário até ao ensino superior, sendo que a maioria foram de professores do ensino secundário.

Na segunda secção dos questionários, os utilizadores foram questionados sobre a sua opinião acerca da utilização de jogos no ensino e costumavam utilizar este recurso no seu material didático. A secção era composta por cinco questões em que os resultados primeira questão são apresentados através do gráfico da Figura 6.3. Dez dos vinte e um concordavam na utilidade dos jogos educativos, no entanto só se os mesmos estivessem bem estruturados e apenas dois utilizadores achavam que os jogos não são úteis pois gerará distrações nos alunos.

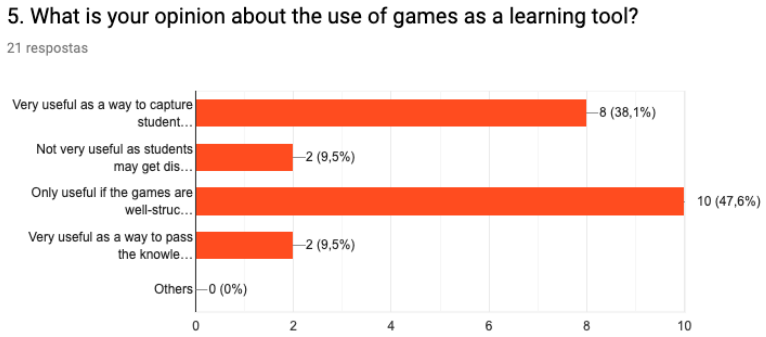


Figura 6.3: Opinião dos utilizadores sobre a utilização de jogos no ensino

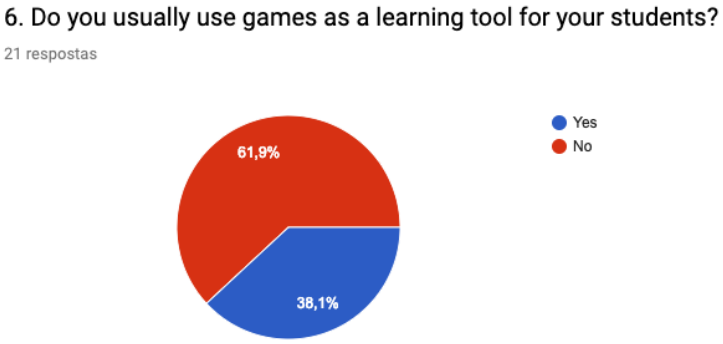


Figura 6.4: Utilização de jogos como método de ensino

Apesar das opiniões positivas sobre a utilização de jogos no ensino, a maioria dos questionados não costuma utilizar como ferramenta para os seus alunos, demonstrado na Figura 6.4, e apenas dois criaram os seus próprios jogos, sendo o Moodle [32] a plataforma em comum para desenvolvimento dos mesmos.

Com base na opinião dos utilizadores podemos concluir que existe interesse neste tipo de material para ser usado na sala de aula, no entanto por desconhecimento das plataformas ou por dificuldade no desenvolvimento de jogos são opções que não são muito usadas pelos professores para criarem os jogos de acordo com o nível e área de ensino das suas turmas.

6.2 Interface e componente gráfica para criação de fluxos de cenários

Durante a terceira e quarta secções do questionário os utilizadores avaliam a interface que testaram em termos de usabilidade, funcionalidades e organização. Numa escala de 1 a 5, os utilizadores dividiram as suas opiniões entre o valor 3 e o valor 4, o que indica que apesar das melhorias que poderiam ser realizadas na plataforma, a aplicação é acessível

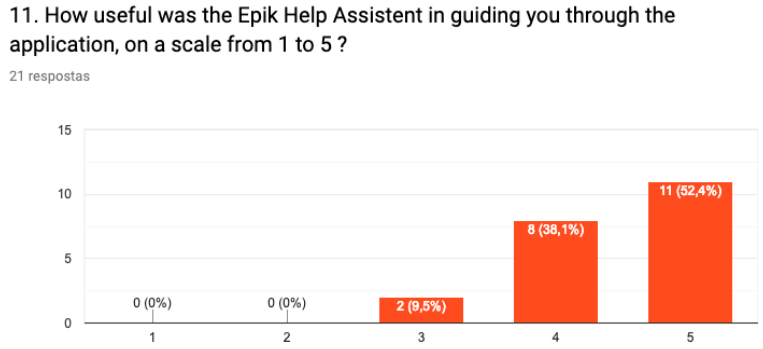


Figura 6.5: Utilidade do manual integrado na aplicação

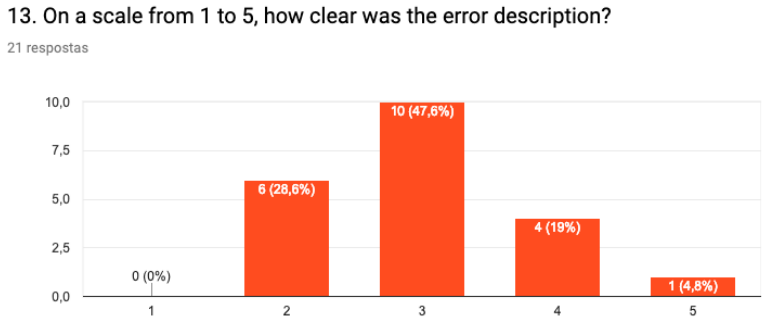


Figura 6.6: Clareza da descrição de erros

em termos de utilização e algo facilitou o uso da plataforma foi o manual integrado existente que, quando questionados sobre o mesmo, a maioria dos utilizadores consideraram o valor máximo da escala em termos de utilidade, ajudando a navegação pela plataforma 6.5.

No momento de geração do jogo, os resultados não foram tão positivos como desejávamos, existiu uma percentagem considerável, no gráfico 6.6, que terminou a geração de um jogo sem sucesso e consequentemente os resultados sobre a descrição dos erros foram afetados onde seis utilizadores consideram a descrição dos erros pouco clara, não ajudando na resolução do problema com que se depararam. Assim sendo, com estes resultados, a descrição dos erros tornou-se um dos principais pontos a melhorar na plataforma de modo a diminuir a percentagem de falhas no momento de geração do jogo.

No geral, a maioria os utilizadores estava satisfeitos com a organização apresentada na plataforma no geral, na estrutura composta por cenários, com a diversidade de transições criadas para esta implementação da aplicação e a diversidade de recursos que são possíveis adicionar aos cenários que compõem o projeto, em que todas estas questões obtiveram a maioria das respostas com o nível de satisfação 4 na escala até 5.

Para a componente gráfica os resultados voltaram a ser positivos, no entanto alguns utilizadores apresentaram alguns pontos a melhorar relativo à posição dos nós do grafo

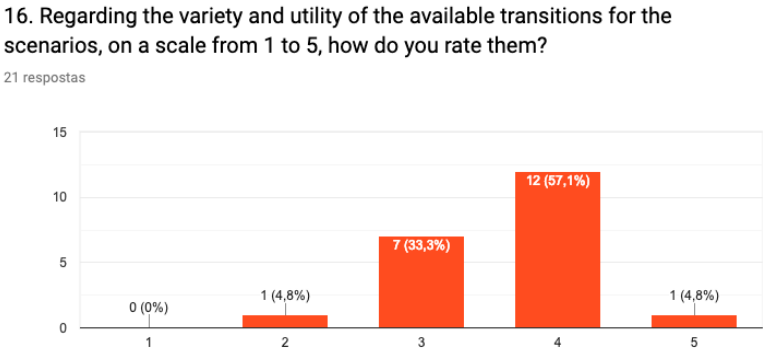


Figura 6.7: Diversidade das transições

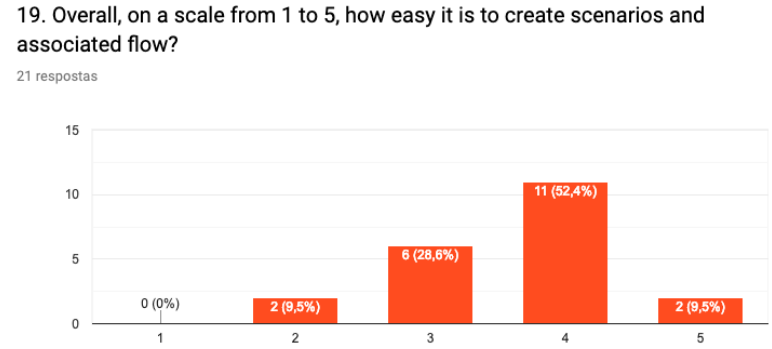


Figura 6.8: Facilidade na criação de cenários e respetivo fluxo

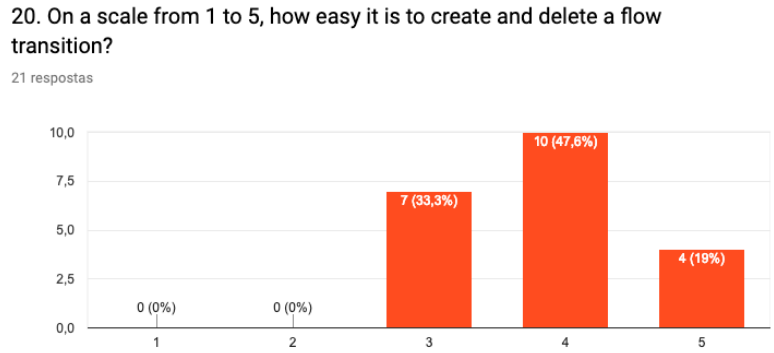


Figura 6.9: Facilidade na criação e eliminação de transições

no momento de criação de transições, pois o grafo quando é atualizado perde as posições dos nós definidas pelo utilizador podendo causar alguma frustração quando o utilizador interage com a componente, como pode ser visível nos graficos 6.8 e 6.10. Porém, as funcionalidades de criação e eliminação de transições foram consideradas fáceis de usar na questão 6.9 por todos os utilizadores sem terem recebido nenhuma avaliação negativa.

Nesta secção é possível apurar alguns pontos a melhorar num trabalho futuro sobre a plataforma em que as mensagens de erro deveriam ser mais claras e o grafo deveria guardar a disposição determinada pelo utilizador caso o *layout* pré-definido não seja o

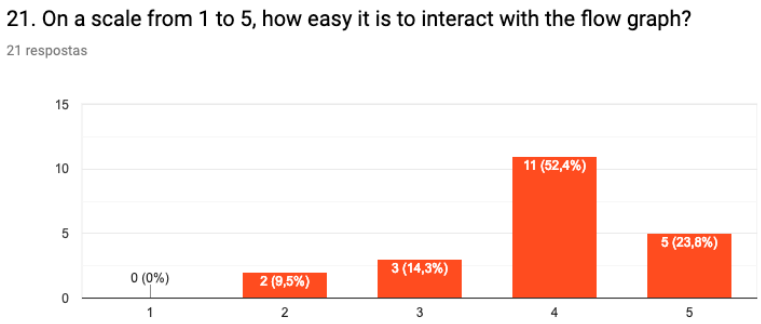


Figura 6.10: Facilidade de interação com a componente

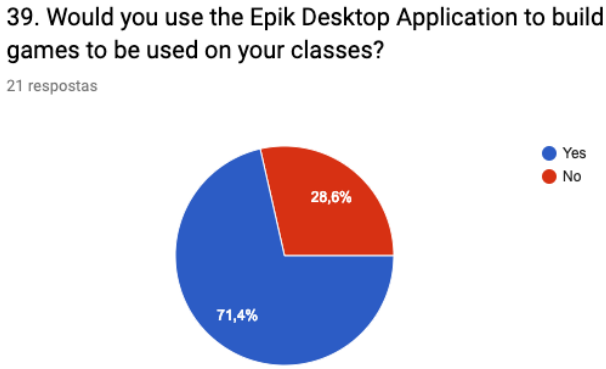


Figura 6.11: Opinião acerca da utilização da aplicação na sala de aula

adequado para o mapa do jogo.

6.3 Satisfação geral

No geral, após os testes e respostas aos questionários, os utilizadores mostraram-se satisfeitos com o desempenho da plataforma e mais de setenta por cento considerariam utilizar o Epik nas suas aulas quer seja, em aulas teóricas, aulas prática, trabalhos de casa ou como método de ensino remoto. Na pergunta 41 foi pedido aos utilizadores para selecionar que palavras usariam para descrever a aplicação do Epik em que as respostas mais comuns foram: amigável, útil, simples e confuso.

Por fim, foi perguntado se recomendariam a aplicação a outros colegas seus interessados no desenvolvimento de jogos educativos para aplicar nas suas aulas e se tinham alguma sugestão ou comentário a fazer. Assim sendo, dois terços dos utilizadores dizem que recomendavam a plataforma aos seus colegas e alguns dos comentários/sugestões feitas foram:

- "A organização do grafo de cenários não é guardada.";

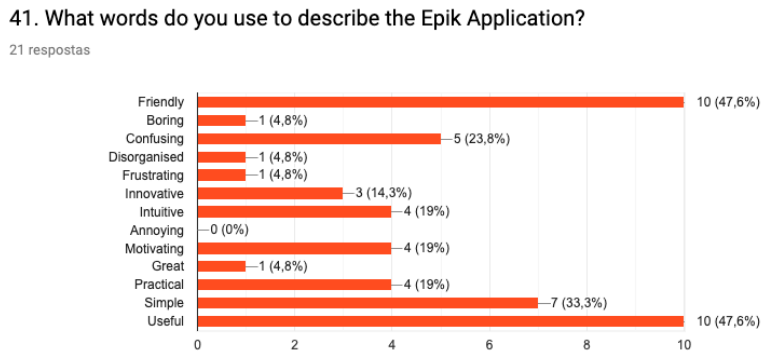


Figura 6.12: Palavras para descrever a aplicação Epik

- "Existem ainda muitos erros na manipulação dos recursos no cenário (posição).";
- "A ideia é muito útil, no entanto nunca consegui criar um jogo sem erros. Seria bom ter um jogo para ficar com uma ideia mais clara. Só entendi o que seria um jogo através das ajudas. Mas foi difícil de usar."

6.4 Sumário

Ao longo desta secção foi possível apurar resultados geralmente positivos sobre a aplicação testada e a utilização de jogos no ensino. Os utilizadores durante a realização dos testes mostraram-se interessados na utilização de jogos como materiais auxiliares às suas aulas, apesar de poucos o costumarem utilizar, pois grande parte afirma que para a sua utilização ser eficaz, o jogo necessita de estar bem estruturado ou os alunos perderão o foco no objetivo principal do jogo. Podemos concluir também que são poucos os utilizadores que já criaram os seus próprios jogos e em ambos os casos a plataforma utilizada foi o Moodle para desenvolver questionários.
Apesar de algumas falhas apontadas à interface comum da aplicação, como a clareza dos erros no momento de geração do jogo, as respostas obtidas refletem a satisfação dos utilizadores durante a utilização da plataforma, sendo uma minoria as avaliações negativas que obtivemos em relação à usabilidade e organização da aplicação. As avaliações positivas obtidas foram influenciadas pelo manual integrado na aplicação *desktop*, que permitiu uma explicação passo a passo da aplicação sem que fosse necessário uma leitura extensa de documentação e a sua avaliação obteve os resultados mais positivos ao longo de todo o questionário.
Na componente gráfica para construção de fluxos foi avaliada a diversidade das transições implementadas, a facilidade de interação com o grafo e a facilidade de adição ou remoção de transições entre os cenários. Os utilizadores consideraram a criação do grafo e sua interação, geralmente boa mas, a existência de uma falha na componente faz com que existam resultados pouco positivos em algumas das avaliações. A falha consistia no

momento de atualização do grafo, pois este não guarda as posições dos nós definidas pelos utilizadores o que poderia causar alguma confusão. Porém, devido à variedade aumentada de transições possíveis de adicionar e à facilidade deste mecanismos de adição e remoção a avaliação da componente é positiva.
No final do questionário foi pedido aos utilizadores que sugerissem ou fizessem comentários à aplicação e após serem identificados os erros que causaram mais problemas durante os testes, estes foram adicionados como sugestão no capítulo seguinte onde são feitas as conclusões e apresentadas ideias para um trabalho futuro sobre a plataforma.

Capítulo 7

Conclusões e trabalho futuro

7.1 Conclusões

Para esta dissertação foi proposto que fosse reformulada e estendida uma plataforma já criada na Faculdade de Ciências e Tecnologia, a plataforma Epik. Esta plataforma, agora com uma componente em formato de aplicação *desktop* possibilitando assim a sua utilização sem necessidade de ligação à internet, permite aos utilizadores/professores o desenvolvimento de jogos educativos sem que seja necessário conhecimentos de programação através de uma interface gráfica, motivando os jogadores para a educação através de jogos individuais ou colaborativos. Os jogos na sua composição têm cenários ligados entre si através de um mecanismo de transições, cujo no conteúdo pode ser formas, recursos multimédia e atividades. Para além das atividades do tipo questionário foram também implementadas as atividades de puzzle, permitindo assim uma maior diversidade no momento de construção do jogo e menor monotonia por parte dos jogadores no momento de execução do jogo.

Duas das limitações mais preponderantes presentes na versão anterior da plataforma consistia no mecanismo utilizado para construir o fluxo de jogo e as transições existentes. Para colmatar essas fragilidades da plataforma, sendo o tema principal desta dissertação, foi construida uma componente gráfica que permite a construção do fluxo de cenário com auxilio de grafos e implementadas novos tipos de transição que permitirão aos jogadores seguirem o seu próprio fluxo de jogo independente dos outros jogadores e com base no seu desempenho no cenário.

Ao longo da implementação da plataforma, tivemos algumas dificuldades devido aos mecanismos de paralelismo apresentados pelo Electron e pelo SQLite que por vezes faziam com que não obtivéssemos os resultados esperados das pesquisas à base de dados, resultando assim em conflitos e inconsistências na plataforma. Também, na discussão

dos modelos de dados e exportação de jogos tivemos problemas que fizeram com que houvessem várias mudanças neste âmbito ao longo do projeto, fazendo assim com que existissem atrasos na implementação da plataforma, mais concretamente com o servidor de execução. Inicialmente tínhamos um ficheiro JSON que serviria de copia da base de dados de um projeto no ambiente de desenvolvimento e que seria importado para o servidor de execução, copiando os seus valores para a base de dados MySQL existente no servidor então sempre que existia uma modificação numa delas tínhamos que manter a coerência entre as duas. Porém, chegamos à conclusão que seria um desperdício de recursos ter uma cópia de algo que já existia em formato JSON enquanto poderíamos ler diretamente do ficheiro durante a execução do jogo evitando assim pedidos à base de dados. Por fim, no momento de geração de ficheiros executáveis verificamos que nem sempre a execução do executável correspondia ao que estava implementado e em desenvolvimento, sendo necessário implementar permissões extra, corrigir o formato dos caminhos e gerar ficheiros para diferentes sistemas operativos. Este tipo de dificuldades fez com que a fase de avaliação da aplicação fosse constantemente adiada deixando pouca margem para uma avaliação mais profunda da aplicação.
Nas testes de avaliação realizados, os utilizadores mostraram-se satisfeitos com a utilidade e simplicidade da aplicação, mas também foram identificados vários pontos a melhorar que foram adicionados à secção seguinte de trabalho futuro a realizar sobre a plataforma.

7.2 Trabalho futuro

Para terminar este capitulo são apresentadas nesta secção algumas melhorias que poderão ser feitas na plataforma num trabalho futuro com vista a consolidar e melhorar o desempenho da mesmo, assim como seria a implementação de novos tipos de jogos, novas propriedades, verificações sobre projetos e novas funcionalidades sobre o fluxo de cenários.
Referindo primeiro a aplicação *desktop*, penso que seria importante realizar um teste de desempenho da plataforma de modo a obter requisitos mínimos para uma utilização eficaz. Para além disso, existem funcionalidades a melhorar na atual implementação de modo a tornar a utilização mais intuitiva e para tal penso que algum do trabalho futuro sobre a plataforma passará por:

- Criar uma lista com os requisitos mínimos do sistema, identificar as funcionalidades mais exigentes e melhorar a performance das mesmas;
- Identificar momentos em que existe paralelismo apresentado pelo Electron e pelo SQLite3, de modo a compreender, analisar e implementar uma prevenção de erros causados em pontos fulcrais da criação do projeto e geração do jogo;
- Análise sobre a complexidade temporal e espacial das funções executadas sobre a aplicação;

- Modificação da interface de modo a ser adaptável a várias resoluções de ecrãs;
- Criação de *templates* de projetos e cenários incluídos na instalação da aplicação e fornecidos no momento de criação dos respetivos;
- No momento de criação do projeto, definir nível e área de ensino permitindo assim uma melhor organização das *dashboards*;
- Permitir cópia de projetos na *dashboard*;
- No momento de construção do projeto evitar pedidos à base de dados de modo a melhorar o desempenho da aplicação e guardando todas as propriedades do projeto apenas quando o utilizador efetua a ação de "Guardar projeto";
- Botões que permitam ao utilizador desfazer/refazer a ultima ação realizada e rotação de elementos;
- Na componente de construção do fluxo de cenários guardar a posição que o utilizador definiu para os nós e apresentar uma lista de *layouts* pré-definidos que permita ao utilizador escolher a que melhor se adapta ao seu fluxo de cenários;

Por parte do servidor de execução seria interessante efetuar testes de capacidade de modo a apurar a longevidade da plataforma após criação de vários jogos, utilizadores e recursos. Para além de novas funcionalidades, existem alguns problemas que foram aparecendo ao longo da implementação do servidor que podem levar a uma melhor utilização da plataforma:

- Permitir o acesso ao servidor por parte de utilizadores e jogadores sem acesso à rede da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa;
- Definir limite de número e tamanho de recursos permitidos no servidor, evitando sobrecarga do servidor ou importações falhadas de jogos por falta de espaço;
- Criar um repositório com jogos públicos, partilhados com todos os utilizadores registados, sendo possível pesquisar por nível e área de ensino, permitindo aos criadores de jogos tirar ideias e discutir sobre como tornar o jogo melhor e mais interativo;
- Permitir uma inserção de uma lista de emails pré definida no momento de importação do jogo sem que seja necessário inserir os emails um a um;
- Envio de notificações quando a data final de um jogo está a chegar ao fim;
- Durante a execução de um jogo *singleplayer* permitir que o jogador coloque o jogo em pausa;
- No mapa de jogo, guardar a posição dos nós definida pelo jogador e apresentar quais os tipos de transição realizados para além dos cenários realizados;

- Implementar os registos para jogos *multiplayer*, com uma maior diversidade, permitir ao utilizador definir que registos deseja para o seu jogo e representação dos registos em gráficos para uma melhor visualização;

Com todos estes pontos a melhorar, penso que no futuro pudesse ser desenvolvido um fórum a incorporar o site do Epik onde os utilizadores registados poderiam trocar ideias, avaliar jogos e dar sugestões, criando assim uma comunidade e promovendo a entreajuda entre utilizadores, jogadores e programadores da plataforma.

Bibliografia

[1] Scirra. *Construct*. URL: https://www.scirra.com/ (acedido em 07/01/2018).

[2] U. Technologies. *Unity*. URL: https://unity3d.com/pt (acedido em 07/01/2018).

[3] Y. Games. *Game Maker: Studio*. URL: https://www.yoyogames.com/gamemaker/ (acedido em 07/01/2018).

[4] P. C.C. L. de Moura Mota. "Jogos no ensino da matemática". Tese de mestrado. Universidade Portucalense Infante D. Henrique, set. de 2009.

[5] B. E. N. Sampaio. "Epik: Plataforma para Desenvolvimento de Jogos para Aprendizagem Colaborativa e Interativa". Tese de mestrado. Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, mai. de 2013.

[6] *Puzzle Figure*. URL: http://www.ducksters.com/games/power_blocks.php (acedido em 17/01/2018).

[7] *Memory Game Figure*. URL: https://www.neok12.com/games/memory/memory.htm (acedido em 17/01/2018).

[8] D. de Lima Feitosa e Fábio Paraguaçu Duarte da Costa. "Aprendizagem Colaborativa e Aplicações". Em: (). URL: https://www.researchgate.net/profile/Douglas_Feitosa/publication/228834420_Aprendizagem_Colaborativa_e_Aplicacoes/links/55e847a108ae65b638997859/Aprendizagem-Colaborativa-e-Aplicacoes.pdf.

[9] P. Dillenbourg. *Collaborative Learning: Cognitive and Computational Approaches*. Advances in learning and instruction series. Elsevier Science & Technology Books, 1999. ISBN: 9780080430737. URL: https://books.google.pt/books?id=Zq4wQgAACAAJ.

[10] *Wisconsin's Guiding Principles for Teaching and Learning*. URL: https://dpi.wi.gov/sites/default/files/imce/cal/pdf/guiding-principles4.pdf (acedido em 23/01/2018).

[11] *Slack*. URL: https://slack.com/.

[12] *Skype*. URL: https://www.skype.com/pt/.

[13] T. Hussain e S. Coleman. *Design and Development of Training Games: Practical Guidelines from a Multidisciplinary Perspective*. Cambridge University Press, 2014. ISBN: 9781316342770. URL: https://books.google.pt/books?id=zTTSBgAAQBAJ.

[14] *MySQL*. URL: https://dev.mysql.com/doc/ (acedido em 19/03/2019).

[15] G. Rauch. *Socket.IO*. URL: http://sockect.io (acedido em 20/03/2019).

[16] *NodeJS*. URL: https://nodejs.org/en/docs/ (acedido em 22/01/2018).

[17] B. Sampaio, C. Morgado e F. Barbosa. "Building Collaborative Quizzes". Em: (2013). DOI: http://dx.doi.org/10.1145/2526968.2526985.

[18] Facebook. *React Desktop*. URL: http://reactdesktop.js.org/ (acedido em 13/11/2017).

[19] GitHub. *Electron*. URL: https://electronjs.org/ (acedido em 12/11/2017).

[20] *Xojo*. URL: https://www.xojo.com/ (acedido em 10/11/2017).

[21] *GraphViz*. URL: https://graphviz.gitlab.io/ (acedido em 10/11/2017).

[22] *DOT*. URL: https://graphviz.gitlab.io/_pages/pdf/dotguide.pdf (acedido em 20/11/2017).

[23] *Prefuse*. URL: http://prefuse.org/ (acedido em 20/11/2017).

[24] *GTAD*. URL: http://gtad.sourceforge.net/ (acedido em 22/11/2017).

[25] M. Franz, C. T. Lopes, G. Huck, Y. Dong, O. Sumer e G. D. Bader. "Cytoscape.js: a graph theory library for visualisation and analysis". Em: (2015). DOI: https://doi.org/10.1093/bioinformatics/btv557.

[26] *Visualgo*. URL: http://visualgo.net/pt (acedido em 22/11/2017).

[27] *SQLite*. URL: https://www.sqlite.org/docs.html (acedido em 19/03/2019).

[28] *Material-UI*. URL: https://material-ui.com// (acedido em 20/01/2018).

[29] GitHub. *axios - Promise based HTTP client for the browser and node.js*. 2019. URL: https://github.com/axios/axios.

[30] M. web docs. *File Reader*. 2019. URL: https://developer.mozilla.org/pt-PT/docs/Web/API/FileReader.

[31] M. Mamede. "Apresentação - Cap. V Ordenação Topológica e Teste à Aciclicidade". 2016.

[32] *Moodle*. URL: https://moodle.fct.unl.pt/ (acedido em 20/03/2019).

ANEXO I

Questionário

I.1 Epik Games Development Application

- User Information

  1. What is your gender?
  2. How old are you?
  3. How many years have you been a teacher?
  4. What educational grade are you teaching?

- Games in Education

  5. What is your opinion about the use of games as a learning tool?
  6. Do you usually use games as a learning tool for your students?
  7. If yes, in which way?
  8. Have you ever used an application to create games as learning tool for your students?
  9. If yes, which applications have you used?

- Epik Interface

  10. On a scale from 1 to 5, how easy to use was the Application interface?
  11. How useful was the Epik Help Assistent in guiding you through the application, on a scale from 1 to 5 ?
  12. When generating a game, have you ever finished the process without success?

13. On a scale from 1 to 5, how clear was the error description?
14. Overall, on a scale from 1 to 5, how do you evaluate the interface organisation of the Epik Desktop Application?

- Epik Games - Scenarios and Flow

15. On a scale from 1 to 5, how do you rate the Epik games organization in scenarios?
16. Regarding the variety and utility of the available transitions for the scenarios, on a scale from 1 to 5, how do you rate them?
17. On a scale of 1 to 5, how do you rate the insertion/removal of resources in game scenarios?
18. On a scale from 1 to 5, how do you found the existing resources types (image, video, pdf and audio) in game scenarios useful ?

- Epik Games - Flow Component

19. Overall, on a scale from 1 to 5, how easy it is to create scenarios and associated flow?
20. On a scale from 1 to 5, how easy it is to create and delete a flow transition?
21. On a scale from 1 to 5, how easy it is to interact with the flow graph?

- Epik Games - Puzzle Activities

22. How easy was to develop the puzzle activities, on a scale from 1 to 5?
23. On a scale from 1 to 5, how useful do you think the inclusion of puzzle activities will be for games?
24. On a scale from 1 to 5, how do you evaluate the diversity of educational subjects that the 3 types of puzzle activities may provide?
25. Regarding the 3 types of puzzles available, how do you qualify each one of their development interfaces?

- Epik Games - Questions Activities

26. On a scale from 1 to 5, what is the probabilistic degree of using quizzes as educational activities in games?
27. Do you think that having 5 types of questions is enough to include a wide range of education subjects?
28. On a scale from 1 to 5, how easy was to develop questions activities?
29. Do you think that the tools and resources available to construct questions are enough to develop these type of activities?

- Epik Games - Collaboration and Helps

30. Overall, on a scale from 1 to 5, how do you rate the available collaboration mechanisms between players in the Epik Games?
31. Do you found the controlled chat, as a collaboration mechanism in an educational game, useful?
32. Do you found the free chat, as a collaboration mechanism in an educational game, useful?
33. How do you rate the notifications about player actions useful, on a scale from 1 to 5?
34. Do you found the display of player moods during an Epik game useful?
35. How do you rate, on a scale from 1 to 5, the utility of the hints in game?
36. How do you rate, on a scale from 1 to 5, the use of context resource in a game?
37. How do you rate, on a scale from 1 to 5, the collaboration between players in the Epik games?

- Satisfaction with the Epik Desktop Application

38. On a scale from 1 to 5, what is your degree of satisfaction with the Epik Desktop Application?
39. Would you use the Epik Desktop Application to build education games to be used on your classes?
40. If yes, in which way?
41. What words do you use to describe the Epik Application?
42. Would you recommend the Epik Application to your friends / colleagues?
43. If you have any suggestions, comments, or review please write them below.

Tiago André Gonçalves Castanho

Plataforma Epik - Componente gráfica e interativa para criação de fluxos de cenários

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Julho, 2019

Tiago André Gonçalves Castanho

Plataforma Epik - Componente gráfica e interativa para criação de fluxos de cenários

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Julho, 2019

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Tiago Castanho 2019
Plataforma Epik - Componente gráfica e interativa para criação de fluxos de cenários 2019